A Convergence Theorem for Graph Shift-type Algorithms

Xuhui Fan, Longbing Cao

Advanced Analytics Institute, University of Technology, Sydney

Abstract

The Robust Graph mode seeking by Graph Shift [1] (RGGS) algorithm represents a recent promising approach for discovering dense subgraphs in noisy data. However, there are no theoretical foundations for proving the convergence of the RGGS algorithm, leaving the question as to whether an algorithm works for solid reasons. In this paper, we propose a generic theoretical framework consisting of three key Graph Shift (GS) components: the simplex of a generated sequence set, the monotonic and continuous objective function and closed mapping. We prove that the GS-type algorithms built on such components can be transformed to fit Zangwill's theory, and the sequence set generated by the GS procedures always terminates at a local maximum, or at worst, contains a subsequence which converges to a local maximum of the similarity measure function. The framework is verified by theoretical analysis and experimental results of several typical GS-type algorithms.

Keywords: Convergence Proof, the Robust Graph mode seeking by Graph Shift

Email addresses: xuhui.fan@student.uts.edu.au (Xuhui Fan), longbing.cao@uts.edu.au (Longbing Cao)

(RGGS) Algorithm, Dominant Sets, Pairwise Clustering, Zangwill's theory.

1. Introduction

The Robust Graph mode seeking by Graph Shift (RGGS) algorithm [1, 2, 3, 4, 5, 6, 7] is a newly proposed algorithm in seeking dense subgraphs (also known as graph mode) and has received much attention in the machine learning and data mining areas. Due to its tremendous advantages in removing the noisy data in learning dense subgraphs, it is popularly used for image processing such as common pattern matching [7, 2], computer vision such as object tracking [3, 4, 5, 6], cluster analysis [1], etc. In addition, its low computational and memory complexity requirements make it feasible for handling real applications, especially for large-scale data. However, little work has been done to build the theoretical foundation for the RGGS algorithm. This leaves important questions about why such algorithms work and whether there is any underpinning foundation to ensure the empirical demonstrations work well for any situations.

The RGGS algorithm originated from the Dominant Sets and Pairwise Clustering (DSPC) algorithm [8, 9], which treats the dense subgraphs discovery problem as a constrained optimization problem and gives a clear definition of the so-called "dominant set", i.e., dense subgraphs. Further, by modifying the existing DSPC (known as Replicator Dynamics in the RGGS algorithm) procedure, the RGGS algorithm adds a "Neighborhood Expansion" procedure to reinforce the learning results. By iteratively employing the DSPC procedure and the newly added Neighborhood Expansion procedure, the RGGS algorithm claims to find a local

maximum of the constraint objective function after a finite number of iterations and further, empirically demonstrates the claim.

However, to the best of our knowledge, none of the existing arguments were built on a solid theoretical foundation with a proper justification of the objective functions' behavior during the procedures and the stopping criteria. All of the above issues are closely related to one thing: the convergence property of the RGGS algorithm. That is to say, we need to ensure that the generated sequence set is convergent or at least contains a convergent subsequence set. It is certainly crucial to have a theoretical analysis of the convergence behavior of the RGGS algorithm before we can confidently utilize it.

Building a proper convergence theorem for the respective learning algorithms is a very important theoretical issue in building solid learning theories. While this is often very challenging, several of existing learning theories include such a component, including the convergence theorem for algorithms with an iterative sequence set. The convergence proof for the fuzzy c-means algorithm (FCM) was provided by Bedzek [10, 11], who employed Zangwill's theory [12, 13] to establish the sequence's convergence property. Hoppner [14] proved the convergence of the axis-parallel variant of the Gustafson-Kessel's algorithm [15] by applying Banach's classical contraction principle [16], which is the general case of FCM. Groll [17] used the equivalence between the original and reduced FCM criteria, and conducted a new and more direct derivation of the convergence properties of FCM algorithms. In addition, Selim [18] treated the k-means clustering problem as a nonconvex mathematical program and provided a rigorous proof of the finite

convergence of the K-means-type algorithms.

The FCM's convergence theory seems complete and this inspires the possibility of it being applied to other purposes, e.g. the RGGS algorithm with an iterative set. Unfortunately, it is not easy to capture the RGGS algorithm's complex characteristics in the convergence proof. To address this problem, we provide a theoretical analysis of the RGGS algorithm. We start with an understanding of the principal characteristics of the RGGS algorithm by breaking it down into three key components, the generated sequence set, the objective function and mapping, and then propose a framework to map such components to the conditions required in Zangwill's theory. We find that the mapped RGGS algorithm can then perfectly match with the key requirements in Zangwill's theory. The convergence theorem for the RGGS algorithm is then developed.

Further, a definition of the so-called "GS-type algorithms" is then given to provide us with a general framework to fit algorithms with similar properties. More importantly, we build up a systematic learning process for them by analyzing the objective functions' behaviors and observing how they arrive at the final results. We illustrate the proposed convergence theorem in terms of proving two typical GS-type algorithms: the RGGS algorithm in [1] and the DSPC algorithm in [9]. The theoretical analysis is then verified against the experimental results.

In summary, this work makes the following substantial contributions:

- A theoretical framework is proposed to analyze the convergence behaviors of the RGGS algorithm. This enables the intrinsic key characteristics embedded in the RGGS algorithm to be effectively captured.

- Taking the two typical GS-type algorithms as examples, we prove the RGGS algorithm and the DSPC algorithm either terminate at a local maximum value or at least contain a subsequence which converges to a local maximum.

- A convergence proof framework is built to generalize the proposed theoretical framework to other GS-type algorithms, namely the RGGS algorithm with similar properties.

The paper is organized as follows. Section 2 introduces the principle of the RGGS algorithm. In Section 3, we first introduce Zangwill's theory, and then propose a framework for extracting three key components in the RGGS algorithm, which are then mapped to the properties of Zangwill's theory . Section 4 discusses the convergence and features of the RGGS algorithm. We extend the convergence proof to other GS-type algorithms and build up a generic convergence proof framework in Section 5. Experiments are conducted in Section 6 to verify the convergence theorem and behaviors. Conclusions and future work can be found in Section 7.

2. Preliminaries

2.1. Rationale of the RGGS Algorithm

The basic principle of the RGGS algorithm is set forth in [1]. From the perspective of graph mining, the RGGS algorithm searches each vertex's dense "nearer" subgraphs with strong internal closeness. Two procedures: Replicator Dynamics and Neighborhood Expansion are recursively employed on each vertex se-

quentially to reach the goal. The former largely shrinks the identified subgraphs, and the latter expands the existing subgraphs, both shifting towards a local graph mode.

In [1, 9], a probabilistic coordinate on Graph G is defined as a mapping: $V \to \Delta^n$, where $\Delta^n = \{\boldsymbol{x} \in R^n : \boldsymbol{x}_i \geq 0, i \in \{1, \cdots, n\}$ and $|\boldsymbol{x}|_1 = 1\}$. The support of $\boldsymbol{x} \in \Delta^n$ is the indices of all non-zero components, denoted as $\delta(\boldsymbol{x}) = \{i|\boldsymbol{x}_i \neq 0\}$, corresponding to a subgraphs $G_{\delta(\boldsymbol{x})}$, and $\boldsymbol{x}_i$ denotes vertex i's occurrence in the subgraphs $G_{\delta(\boldsymbol{x})}$ to some extent.

The algorithm operates on an affinity matrix $A = (a_{ij})^{n \times n}$, in which a_{ij} measures the similarity between vertices i and j. Then, the internal similarity of subgraphs $G_{\delta(\boldsymbol{x})}$ is measured as:

$$g(\boldsymbol{x}) := a(\boldsymbol{x}, \boldsymbol{x}) = \sum_{i,j=1}^{n} a_{ij}\boldsymbol{x}_i\boldsymbol{x}_j = \boldsymbol{x}^T A\boldsymbol{x}. \tag{1}$$

Accordingly, a local maximum solver of $g(\boldsymbol{x})$ can be taken to represent the desired dense subgraphs. The identification of such local maximum regions is equivalent to solving the following quadratic optimization problem:

$$\begin{cases} \text{maximize} & g(\boldsymbol{x}) = \boldsymbol{x}^T A\boldsymbol{x} \\ \text{subject to} & \boldsymbol{x} \in \Delta^n \end{cases} \tag{2}$$

2.2. Mapping Definitions

We state here the problem of the RGGS algorithm and propose a corresponding mapping to form a conceptual understanding of the RGGS algorithm.

In general, the RGGS algorithm defines a mapping $T_m : \Delta^n \to \Delta^n$ to obtain the iterative sequence set as:

$$\boldsymbol{x}^{(k)} = T_m(\boldsymbol{x}^{(k-1)}) = \cdots = (T_m)^{(k)}(\boldsymbol{x}^{(0)}); k = 1, 2, \cdots . \tag{3}$$

where $\boldsymbol{x}^{(0)}$ is an initial starting point, and superscripts in parentheses correspond to the iteration number. Subsequently, the convergence of the RGGS algorithm is to test whether the iterative sequence set $\{\boldsymbol{x}^{(k)}\}_{k=1}^{\infty}$ generated by T_m converges to a local maximum solver of the problem in Equation (2).

The mapping function T_m plays a crucial role in the RGGS algorithm, combining the Replicator Dynamics procedure (B^{m_k}) and the Neighborhood Expansion procedure (C). Therefore, in order to understand and analyze the convergence problem, it is essential to have a deep understanding of the mapping function. Accordingly, T_m is broken down as:

$$T_m := B^{m_k} \circ C. \tag{4}$$

In Equation (4), B^{m_k} represents the k-th ($k \leq m$) Replicator Dynamics procedure; m_k corresponds to the number of transformation B in the k-th Replicator Dynamics procedure when a subgraphs's mode is reached in this procedure (this result is actually a special case of Theorem 3). B is the transformation expressed

as:

$$\begin{aligned} B :&\Delta^n \to \Delta^n, \boldsymbol{x}(l_k) \to \boldsymbol{x}(l_k+1) \\ =&(\frac{\omega_1(l_k)\boldsymbol{x}_1(l_k)}{\sum_{i=1}^n \omega_i(l_k)\boldsymbol{x}_i(l_k)}, \cdots, \frac{\omega_n(l_k)\boldsymbol{x}_n(l_k)}{\sum_{i=1}^n \omega_i(l_k)\boldsymbol{x}_i(l_k)}). \end{aligned} \tag{5}$$

In Equation (5), $\omega_i(l_k) = (A\boldsymbol{x}(l_k))_i = \sum_{j=1}^n a_{ij}\boldsymbol{x}_j(l_k)$, $i \in \{1, \cdots, n\}$, $l_k \in \{1, \cdots, m_k - 1\}$.

C is the Neighborhood Expansion procedure, denoted as:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \Delta\boldsymbol{x} = \boldsymbol{x}^{(k)} + t^*\boldsymbol{b}. \tag{6}$$

Details of t^* and $\boldsymbol{b}$ are explained in Appendix A.

2.3. Detailed Procedure and Stopping Criteria

The RGGS algorithm in [1] is an iterative process for seeking dense subgraphs which starts from each vertex in the graph. The pseudo-codes below illustrate the whole process.

Require: $A^{n\times n}$, the affinity matrix of the whole data set with the diagonal value 0;

$\{\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1^n}\}$, initial starting points, usually taken as $\{\boldsymbol{e} = \{\boldsymbol{e}_i\}_{i=1^n}\}$

1: **for** i=1,...n **do**

2: do Replicator Dynamics (Equation (5)) of $\boldsymbol{x}_i$

3: **if** (result $\boldsymbol{x}$ is the mode of graph) **then**

4: go to Step 11

5: **end if**

6: do Neighborhood Expansion (Equation (6)) of $\boldsymbol{x}_i$

7: **if** (result $\boldsymbol{x}$ is the mode of graph) **then**

8: go to Step 11

9: **end if**

10: **end for**

11: **return** the belonging clusters $\boldsymbol{c} = \{\boldsymbol{c}_i\}_{i=1}^n$ corresponding to the starting points $\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^n$.

The stopping criteria of the algorithm, or the solution set of the problem (Equation (2)), is set to satisfy the Karush-Kuhn-Tucker (KKT) condition [19]:

$$\Gamma := \{\boldsymbol{x} \in \Delta \mid \boldsymbol{x} \text{ satisfies } (A\boldsymbol{x})_i \begin{cases} = \lambda, & i \in \sigma(\boldsymbol{x}); \\ \leq \lambda, & i \notin \sigma(\boldsymbol{x}). \end{cases} \}. \tag{7}$$

Here $(A\boldsymbol{x})_i$ is the i-th component of $A\boldsymbol{x}$; λ is the Lagrange multiplier. $\sigma(\boldsymbol{x}) = \{i \in \{1, \cdots, n\} | \boldsymbol{x}_i \neq 0\}$ corresponds to the subgraphs defined above.

As stated above, the RGGS algorithm is implemented on each vertex's evolving process by recursively using the Replicator Dynamics procedure and the Neighborhood Expansion procedure, until it reaches the desired solution, i.e., satisfying the KKT condition (Equation (7)) to each of the starting vertex.

3. Mapping from the RGGS Algorithm to Zangwill's Theory

Zangwill's theory [12, 13] is fundamental in terms of proving the convergence of iterative sets for its general applicability. Since the process of the RGGS al-

gorithm embeds similar iterative set ideas, our idea here is to build a mapping from the RGGS algorithm's principle to Zangwill's theory. If this works well, then we can use Zangwill's theory to build the convergence system for the RGGS algorithm.

3.1. Zangwill's Theory

Definitions and lemmas are introduced before we present Zangwill's theory.

Definition 1. *A point-to-set mapping Ω from set X to power set Y is defined as $\Omega : X \rightarrow P(Y)$, which associates a subset of Y with each point in X, $P(Y)$ denotes the power set of Y.*

Definition 2. *Given a function f and an element c of the domain I, f is said to be continuous at the point c if the following holds: for every $\varepsilon > 0$, there exists a $\eta > 0$ such that for all $x \in I$, $|x - c| < \eta \Rightarrow |f(x) - f(c)| < \varepsilon$.*

Definition 3. *A point-to-set mapping $\Omega : X \rightarrow P(Y)$ is said to be closed at a point $\boldsymbol{x}^*$ in X if $\{\boldsymbol{x}^{(m)}\} \subset X$ and $\boldsymbol{x}^{(m)} \rightarrow \boldsymbol{x}^*, \boldsymbol{y}^{(m)} \in \Omega(\boldsymbol{x}^{(m)})$ and $\boldsymbol{y}^{(m)} \rightarrow \boldsymbol{y}^*$ imply that $\boldsymbol{y}^* \in \Omega(\boldsymbol{x}^*)$.*

The following Lemma 1 is induced by integrating a continuous function with a point-to-set mapping:

Lemma 1. *Let $C : M \rightarrow V$ be a function and $B : V \rightarrow P(V)$ be a point-to-set mapping. Assume C is continuous at ω^* and B is closed at $C(\omega^*)$, then the point-to-set mapping $A = B \circ C : M \rightarrow P(V)$ is closed at ω^*.*

As the composition of continuous functions is still a continuous function, we have Lemma 2:

Lemma 2. *Given two continuous functions:* $f : I \to J(\subset \boldsymbol{r}), g : J \to \boldsymbol{R}$*, the composition* $g \circ f : I \to \boldsymbol{R}, x \mapsto g(f(x))$ *is continuous.*

Accordingly, Zangwill's theory is described below.

Theorem 1. *Given an algorithm on* X*,* $\boldsymbol{x}^{(0)} \in X$*, assume the sequence* $\{\boldsymbol{x}^{(k)}\}_{k=1}^{\infty}$ *is generated which satisfies*

$$\boldsymbol{x}^{(k+1)} \in \boldsymbol{A}(\boldsymbol{x}^{(k)}) \tag{8}$$

For a given solution set $\Gamma \subset X$ *of an algorithm, if the following three properties holds:*

Compact *The sequence set* $\{\boldsymbol{x}^{(k)}\}_{k=0}^{\infty} \subset S$ *for* $S \subset X$ *is a compact set.*

Decreasing *There is a continuous function* Z *on* X *such that*

1) if $\boldsymbol{x} \notin \Gamma$*, then* $Z(\boldsymbol{y}) < Z(\boldsymbol{x})$ *for all* $\boldsymbol{y} \in \boldsymbol{A}(\boldsymbol{x})$*.*

2) if $\boldsymbol{x} \in \Gamma$*, then* $Z(\boldsymbol{y}) \leq Z(\boldsymbol{x})$ *for all* $\boldsymbol{y} \in \boldsymbol{A}(\boldsymbol{x})$*.*

Closed *The mapping* $\boldsymbol{A}$ *is closed at all points of* $X \backslash \Gamma$*.*

Consequently, either the algorithm stops at the point where a solution is identified or there exists such a k *so that for all* $k + j$ *(*$j \geq 1$*) there is a convergent subsequence of* $\{\boldsymbol{x}^{(i_k)}\}_{k=0}^{\infty}$ *in the solution set* Γ*.*

Zangwill's theory provides a feasible direction to verify an algorithm's convergence behavior, especially for those with iterative processes. Due to its general flexibility, it has been applied widely to prove the convergence of algorithms with similar properties, including clustering and optimization. Of all the iterative algorithms, here we are interested in the RGGS algorithm with the monotonic property.

3.2. Mapping from the RGGS Algorithm to Zangwill's Theory

Considering the conditions in Zangwill's theory, we further break down the GS process and extract the following characteristics from it (detailed verification will be given in Section 4):

1), Simplex of generated sequence set : The candidate solution sequence set $\{\boldsymbol{x}^{(k)}\}_{k=0}^{\infty}$ generated by the mapping T_m lies in $\Delta^n = \{\boldsymbol{x} \in R^n : \boldsymbol{x}_i \geq 0 \text{ and } |\boldsymbol{x}|_1 = 1\}$, i.e., the standard n-simplex of R^n in any step k ($k \leq m$);

2), Monotonic and continuous objective function : The objective function $g(\boldsymbol{x}) = \boldsymbol{x}^T A \boldsymbol{x}$ is continuous and strictly increases during the mapping T_m according to the Propositions 2-4 (see Section 4);

3), Closed mapping : The mapping $T_m = B^{m_k} \circ C$ is closed during each procedure in accordance to Propositions 5-6 (see Section 4) at all points of the generated sequence set.

These three properties are also the key components in an algorithm, that is to say, with these vital feature requirements clearly prescribed, an algorithm is fixed

into a predefined framework, including the generated sequence set defining the scope of the variables, the objective function's behavior describing the algorithm mapping efforts towards the expected goals, and the mapping itself with restricted properties.

The extraction of the above three key features makes it possible to map the RGGS algorithm to Zangwill's theory. Table 1 depicts the one-to-one correspondence to build the mapping between the RGGS algorithm's principle and Zangwill's theory in terms of the key properties embedded in the respective systems.

Table 1: Mapping between the RGGS algorithm and Zangwill's theory

the RGGS algorithm		Zangwill's theory
$Simplex$	$\sim$	$Compact$
$Monotonic$	$\sim$	$Decreasing$
$Closed$	$\sim$	$Closed$

4. Convergence of the RGGS Algorithm

Several propositions are declared before deeply analyzing the convergence behavior of the RGGS algorithm for three RGGS algorithm's components: the generated sequence set, the objective function and mapping. Detailed proofs of these propositions are given in Appendix B-E.

The solution set generated by the RGGS algorithm is compact according to Proposition 1.

Proposition 1. *The sequence set* $\{\boldsymbol{x}^{(k)}\}_{k=0}^{\infty} \subset S$ *generated by the mapping* $T_m = B^{m_k} \circ C$ *is a compact set.*

Propositions 2-4 discuss the monotonicity of $g(\boldsymbol{x})$ under the mapping $T_m = B^{m_k} \circ C$, considering the monotonicity of the RGGS algorithm's main properties.

Proposition 2. *The objective function* $f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}$ *strictly increases along any non-constant trajectory of Equation (5) when* $\boldsymbol{x} \in X/\Gamma$.

Proposition 3. *The objective function* $f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}$ *strictly increases along the neighborhood expansion operation in Equation (6).*

Proof. It can be derived from the definition of $\Delta\boldsymbol{x}$ in Appendix B. □

Proposition 4. $g(x) = \boldsymbol{x}^T A\boldsymbol{x}$ *is a function that is both continuous and strictly increasing during the mapping* $T_m = B^{m_k} \circ C$ *when* $\boldsymbol{x} \in X/\Gamma$*, but just increasing if* $\boldsymbol{x} \in \Gamma$.

Propositions 5-6 validate the closed mapping property of the RGGS algorithm.

Proposition 5. *The mapping* C *is closed on all points of* $X \backslash \Gamma$.

Proposition 6. *The mapping* $T_m = B^{m_k} \circ C$ *is closed on* X/Γ.

Proof. According to the definition of B (Equation (5)), it is continuous on X/Γ. C is closed on X/Γ as per Proposition 5. According to Lemma 1, T_m is closed on X/Γ. □

With the above preparations, we have the following Theorem 2.

Theorem 2. *Let $A = (a_{ij})^{(n\times n)}$ be a affinity matrix with diagonal values 0, T_m, Γ be defined as Equation (4), Equation (7), and $\boldsymbol{x}^{(0)}$ be an arbitrary initial starting point, then either the iteration sequence $\{\boldsymbol{x}^{(r)}\}$ ($r = 1, 2, \ldots$) terminates at a point $\boldsymbol{x}^*$ in the solution set Γ or there is a subsequence converging to a point in Γ.*

Proof. Taking $\hat{g}(\boldsymbol{x}) = -g(\boldsymbol{x}) = -\boldsymbol{x}^T A\boldsymbol{x}$ as the continuous function Z and T_m as the algorithm mapping A in Theorem 1. Proposition 1 shows that the sequence set $\{\boldsymbol{x}^{(k)}\}_{k=1}^{\infty}$ generated by T_m is a compact set. $\hat{g}(\boldsymbol{x})$ is continuous and strictly decreasing as the continuity and strict increasing characteristics of $g(\boldsymbol{x})$ in the trajectory of T_m are proven by Proposition 4. Proposition 6 asserts T_m is closed on x/Γ (Γ is the solution set defined in Equation (7)). According to Zangwill's theory, Theorem 2 holds as all three properties are satisfied. □

This result lays a foundation for theoretically guaranteeing the applicability of the RGGS algorithm to respective applications. It also ensures the RGGS algorithm to reach at least a local maximum of the objective function after a finite number of mapping implementations.

5. Convergence for Other GS-type Algorithms

Here we further expand the proposed RGGS algorithm convergence theorem's proof to other GS-type algorithms. We take the Dominant Sets and Pairwise Clustering (DSPC) algorithm [9] as an example, because it is the original method proposed to implement the RGGS algorithm's idea. The DSPC algorithm shares the same goal as the RGGS algorithm in terms of finding dense subgraphs, i.e.,

dominant sets $\sigma(\boldsymbol{x})$. Their implementations are mostly similar, however they can be differentiated in whether employing the neighborhood expansion or not. The RGGS algorithm in [1] does but the DSPC algorithm does not.

The detailed implementation of the DSPC algorithm can be found in [9], but here we focus on discussing its convergence behavior. The DSPC algorithm consists of three key components and holds properties similar to the RGGS algorithm, while differings in several aspects as discussed below.

1), Simplex of generated sequence set : The sequence set $\{\boldsymbol{x}^{(k)}\}_{k=0}^{\infty}$ generated by the mapping B always lies in Δ^n;

2), Monotonic and continuous objective function : The objective function $g(\boldsymbol{x}) = \boldsymbol{x}^T A\boldsymbol{x}$ is continuous and strictly increasing during the mapping B;

3), Continuous mapping The mapping B is continuous.

Table 2 further discusses the relationships between the DSPC algorithm, the RGGS algorithm and Zangwill's theory.

Table 2: Mapping between GS-type algorithms and Zangwill's theory

the DSPC algorithm		the RGGS algorithm		Zangwill's theory
$Simplex$	$\sim$	$Simplex$	$\sim$	$Compact$
$Monotonic$	$\sim$	$Monotonic$	$\sim$	$Decreasing$
$Continuous$	$\sim$	$Closed$	$\sim$	$Closed$

[20] discusses the convergence behavior of a continuous-time version of the DSPC algorithm, however, its method can not be applied to the case here. The

discrete-time setting of the DSPC algorithm makes it particularly difficult to verify the three properties above. Here some related propositions are first introduced. Then we discuss the convergence property of its discrete-time version by involving Zangwill's theory.

Proposition 7. *The sequence set* $\{\boldsymbol{x}^{(k)}\}_{k=0}^{\infty} \subset S$ *generated by the mapping* B^{m_k} *is a compact set.*

For its proof details, refer to Appendix B.

Proposition 8. *If a mapping* $f : S \to T$ *is continuous on* S*, then* f *is closed on* S*.*

Theorem 3. *Let* $A = (a_{ij})^{(n\times n)}$ *be a affinity matrix with diagonal values 0,* B *be defined as by Equation* (5)*, and* $\boldsymbol{x}^{(0)}$ *be an arbitrary initial starting point, then either the iteration sequence* $\{\boldsymbol{x}^{(r+1)} = B(\boldsymbol{x}^{(r)})\}, (r = 1, 2, \ldots)$ *terminates at a point* $\boldsymbol{x}^*$ *in the solution set* Γ *or there is a subsequence converging to a point in* Γ*.*

Proof. Taking $\hat{g}(\boldsymbol{x}) = -g(\boldsymbol{x}) = -\boldsymbol{x}^T A\boldsymbol{x}$ as the continuous function Z and B as the algorithm mapping A in Theorem 1. Proposition 7 shows that the sequence set $\{\boldsymbol{x}^{(k)}\}_{k=1}^{\infty}$ generated by B is a compact set. $\hat{g}(\boldsymbol{x})$ is continuous and strictly decreasing according to the continuity and strict increase of $g(\boldsymbol{x})$ in the trajectory of T_m, which are proven in Proposition 2. Proposition 8 asserts B is closed on x/Γ, while Γ is the solution set defined in Equation (7). According to Zangwill's theory, Theorem 3 holds. □

The mapping makes the GS-type algorithms sharing similar properties as Zangwill's theory, and further enables the proof of the convergence of GS-type algorithms by following the basic processes in Zangwill's theory. Hence, we here propose a Zangwill's theory-based convergence framework for those algorithms sharing the core GS principles discussed above, called the "GS-type algorithms".

Definition 4. *An algorithm is a GS-type algorithm if and only if it satisfies the conditions on three key components: a simplex of generated sequence set, a monotonic and continuous objective function, and closed mapping.*

We provide a flowchart (Figure 1) to illustrate our proof process in detail. It presents a guideline to prove the convergence of GS-type algorithms step by step.

(1) Break down a GS-type algorithm into three parts: a generated set, an objective function and mapping.

(2) Check if these three parts all satisfy the corresponding requirements. Any part which does not is regarded as unsuitable for applying this framework, otherwise it is convergent.

Of the two processes in Figure 1, the verification of properties of an algorithm is usually the difficult part, especially for the objective function's monotonic behavior. We will specifically discuss the objective function's empirical monotonic behavior in Section 6.

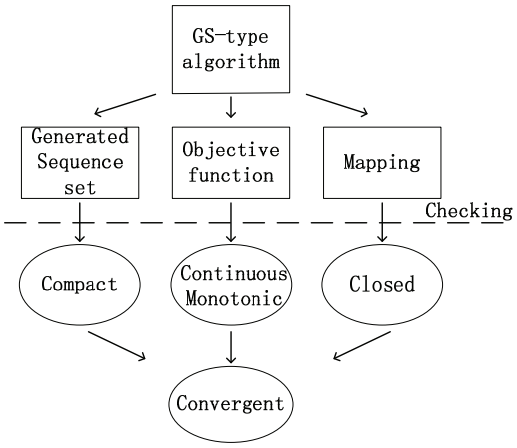

Figure 1: The proof framework for the GS-type algorithm convergence.

6. Experimental Verification

The RGGS algorithm and the DSPC algorithm are all independently implemented in MATLAB2011b. Our experiments are conducted on an Acer Aspire 4720Z laptop having an Intel Pentium DualCore T2330(1.6GHz, 533MHz FSB, 1MB L2 cache), with 2GB DDR2 RAM, using the LINUX operating system.

6.1. Experimental Settings

Since GS-type algorithms manipulate data based on a affinity matrix, we construct a affinity matrix instead of real data sets with its element values uniformly sampled within the interval $[0, 1]$, and the matrix dimensionality scaling from 100 to 3000. Also, we consider cases in which the matrices are fully dense matrices (FDM), partially dense matrices (PDM), and block tridiagonal matrices (BTM) respectively.

The initial starting point $\boldsymbol{x}$ can be randomly chosen or be the single vertex

$\{I_i, i = 1, \cdots, n.\}$. In our experiments, we use the single vertex in the same way as in [7]. We test the RGGS algorithm [1] as well as the DSPC algorithm [9]. Each algorithm runs for three times to obtain an averaged performance. We verify the proposed GS convergence theory through experiments and focus on testing convergence performance. The number of transformations (m_k) in Replicator Dynamics, the whole iteration number (m), running time (T), and average iteration running time ($t = T/m$) are presented to evaluate the convergence performance.

Table 3: Testing results for the RGGS algorithm and the DSPC algorithm

the RGGS algorithm							the DSPC algorithm				
Case	Scale	m_k	m	$T(s)$	$t(s)$	S.R.(%)[1]	Case	Scale	m_k	$T(s)$	S.R.(%)[1]
FDM	100	1228.5	2.52	0.12	0.05	1.0	FDM	100	953.2	0.14	1.0
	500	1297.3	2.96	0.40	0.14	0.2		500	1298.5	0.37	0.2
	1000	1101.7	3.75	1.01	0.27	0.1		1000	1604.2	1.10	0.1
	1500	1401.2	3.24	3.87	1.19	0.0		1500	1469.6	4.25	0.0
	2000	1575.4	3.49	20.00	5.73	0.0		2000	1448.1	18.61	0.0
	3000	1511.4	3.56	50.15	14.09	0.0		3000	1785.3	56.20	0.0
PDM	100	236.3	2.22	0.02	0.01	73.9	PDM	100	203.3	0.02	73.3
	500	285.9	2.24	0.08	0.04	73.8		500	299.6	0.07	73.7
	1000	275.4	2.38	0.20	0.09	73.6		1000	326.7	0.15	73.6
	1500	275.9	2.40	0.94	0.39	73.6		1500	338.2	0.55	73.7
	2000	348.1	2.41	3.32	1.38	73.7		2000	288.8	2.11	73.6
	3000	332.9	2.50	7.75	3.10	73.7		3000	389.3	6.85	73.7
BTM	100	794.9	2.16	0.06	0.03	78.0	BTM	100	717.6	0.05	78.0
	500	1185.3	2.65	0.31	0.12	78.0		500	1177.7	0.28	78.0
	1000	1221.6	2.67	0.68	0.25	78.0		1000	1130.8	0.74	78.0
	1500	1198.6	2.97	2.96	1.00	78.0		1500	1226.5	2.63	78.0
	2000	1289.9	3.02	13.57	4.50	77.9		2000	1417.8	13.24	78.0
	3000	1406.2	3.13	38.55	12.34	78.0		3000	1517.4	38.16	78.0

[1] S.R. refers to the sparse rate, i.e. the proportion of the number of zero elements to the number of whole elements.

6.2. Objective Function Behavior

Figure 2 depicts the behaviors of objective functions in the RGGS algorithm corresponding to three representative vertices under the PDM and 500 scales. The

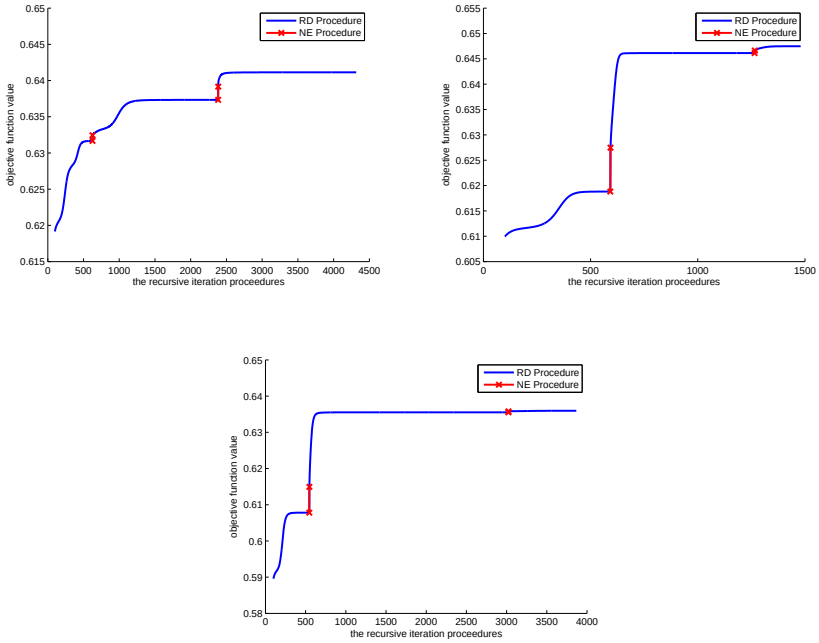

Figure 2: Objective function behaviors for vertices 71 (up left), 231 (up right) and 265 (down).

X-axis represents the times of the evolving process, and the Y-axis stands for the objective function's value. To fairly compare the performance, B and C take the same times. Thus, the rate of evolving times is $m_k : 1$ between Replicator Dynamics and Neighborhood Expansion in the k-th iteration, according to Equation (4).

Observing these three vertices' evolving behaviors, we can see all the three vertices reach a local maximum value. The objective function's value keeps increasing along the evolving process, which is in accordance with the conclusion in Proposition 4. This perfectly matches the outcomes from the theorem we have proposed.

As the curve starts from reflecting the outcomes of the initial process of the Replicator Dynamics, a steep increasing trend is observed in the first few steps followed by a gentle uptrend movement. When the process moves to the Neighborhood Expansion procedure, it produces a huge jump compared to the flat climbing in the Replicator Dynamics. This is because the Neighborhood Expansion procedure is one that pulls the candidate solution from one local dense subgraphs towards the dense graph and the dense value will change greatly when the subgraphs changes.

Also, we can notice that most of the evolving time is related to Replicator Dynamics, denoting that most of the calculation is on the Replicator Dynamics, i.e., searching dense subgraphss.

6.3. Convergence Performance

The convergence performance of the RGGS algorithm and the DSPC algorithm depends on many factors: the scale of the affinity matrix, matrix structure, element value, etc. We test different scenarios by changing the scale and the sparsity of the matrix. The scenarios and results are given in Table 3.

The experimental results presented in the RGGS algorithm and the DSPC algorithm are a little different for their diverse mapping definitions, with the former being $T_m = B^{m_k} \circ C$ and the latter being B.

The results show that both the RGGS algorithm and the DSPC algorithm converge in cases with FDM, PDM and BTM. In each case, the transformation B's number m_k and the iteration T_m's number m increase slowly when a matrix's scale

grows, sometimes even with a small drop, e.g., in the FDM case for the RGGS algorithm, m_k's values in the third row and m's value in the fourth row are both smaller than the previous ones although the matrices' scale increases. What is more, when the matrices's scale is 3,000, which is 30 times the first one's scale in each case, its m_k and m's values are less than 2 times larger. This indicates that the sparse rate of the affinity matrix also plays an important role in the computational cost.

Also, the RGGS algorithm's transformation number, iteration number and running time all decrease when the matrix becomes sparser, the same occurs with the DSPC algorithm's transformation number and running time. This indicates that if we incorporate prior information and set the unrelated vertices' similarity value as 0, we can save much computational cost. However, both of the two algorithms perform worse on the block tridiagonal matrix compared to the partially dense matrices' case, even with a smaller sparse rate. This is due to the fact that the calculation of block tridiagonal matrices is quite similar to the one of the smaller scale with full dense matrices. As shown above, it is heavily computationally loaded.

6.4. Gaussian Clusters with Uniformly Noise

The convergence performance of GS-type algorithms is further tested in the scenarios of Gaussian clustering. More specifically, we generate the synthetic data by using the same routine as in [1], in which clusters are assumed to follow two-dimensional Gaussian distributions evenly located in the $[0, 1] \times [0, 1]$ square,

and noise is represented by uniformly distributed points in this square. The testing scenarios include various numbers of clusters, various sparse rates and various sizes of clusters. In all of the learning scenarios below, we set the noisy points occupy 25% of the whole data set.

6.4.1. Various Numbers of Clusters

In this learning scenario, we set $2, 4, 9, 16$ as the cluster numbers in the four cases. Each of the clusters is assumed to be composed of 100 points. Plots of these 4 scenarios are displayed in Figure 3.

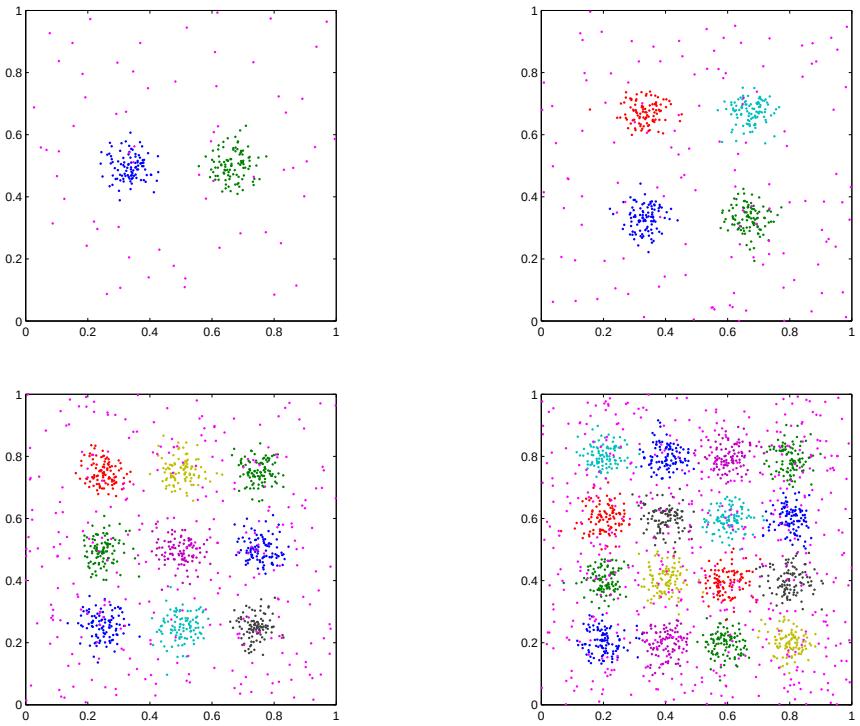

Figure 3: Four cases in the scenario of various numbers of clusters.

Table 4 shows the clustering performance of the RGGS algorithm in these 4

cases. As can be seen, a significant trend is that the clustering performance (Accuracy and NMI) gradually improves while we have more clusters and data points in this unit square. Also, the RGGS algorithm needs less Replicator Dynamic steps within one iteration while more clusters and data points are presented. These phenomena may be due to the reason that more clusters and data points help to better define the structure of the affinity matrix.

Table 4: Various mixture modelling

The RGGS algorithm						
Scale	clu. no.	Accuracy[1]	NMI[2]	m_k	m	$T(s)$
267	2	$0.77(0.01)$	$0.59(0.01)$	2704.11	$2.22(0.21)$	3.37
533	4	$0.86(0.01)$	$0.70(0.01)$	2571.98	$2.23(0.10)$	4.10
1200	9	$0.92(0.00)$	$0.78(0.01)$	1631.43	$2.71(0.62)$	14.62
2133	16	$0.96(0.00)$	$0.83(0.01)$	1445.20	$3.19(0.35)$	31.17

[1] Accuracy refers to the prediction accuracy of data point pair (whether two data points belong to the same cluster). It is calculated as the ratio of the number of the whole data point pairs divided by the number of correctly predicted data point pairs.

[2] NMI refers to the Normalized Mutual Information, which measures the clustering performance. The larger value of NMI, the better of the clustering performance.

6.4.2. Various Sparse Rates of the Affinity Matrix

This learning case has 9 clusters in the unit square (the same as the left bottom one in Figure 3). To promote the sparsity in the affinity matrix, we manually set the value in the affinity matrix to be 0 while it is smaller than a fixed threshold. The thresholds for testing are $[0.2, 0.4, 0.6, 0.8]$ respectively. As can be seen from the results in Table 5, it is interesting to find that the clustering performance is hardly

affected by the increase of the sparse rate, while the running time significantly reduces. Accordingly, we conclude that the clustering results of RGGS algorithms depend mainly on the larger values (e.g., ≥ 0.8) of the affinity matrix. While a high sparse rate would promote fast learning without degrading the performance, we strongly recommend to make the affinity matrix sparse before use.

Table 5: Different sparse rates

The RGGS algorithm						
Threshold	Accuracy	NMI	m_k	m	$T(s)$	S.R.
0.0	0.92(0.00)	0.78(0.01)	1631.43	2.71(0.62)	14.62(2.73)	0.00(0.00)
0.2	0.92(0.00)	0.78(0.00)	1992.52	2.51(0.13)	8.41(1.52)	0.51(0.00)
0.4	0.92(0.00)	0.78(0.00)	1914.58	2.58(0.12)	8.74(1.54)	0.67(0.00)
0.6	0.92(0.00)	0.78(0.01)	1564.33	2.91(0.35)	7.54(0.97)	0.82(0.00)
0.8	0.93(0.00)	0.78(0.01)	1484.72	2.95(0.53)	6.85(0.78)	0.91(0.00)

6.4.3. Various Sizes of Clusters

The general setting of this case study remains the same as the previous one. Instead of changing the sparse rate, we here change the size of the central clusters (the "purple" clusters in the right bottom part of Figure 3). The central cluster size is set to be $[50, 100, 200, 300, 400, 500]$ sequentially. As can be seen in Table 6, the RGGS algorithm does not perform well in the "highly" imbalanced data case. Both of the accuracy and NMI score decreases with the increase of the central cluster size.

Table 6: RGGS performance in the case of various cluster sizes

The RGGS algorithm					
Center size	Accuracy	NMI	m_k	m	$T(s)$
50	0.91(0.01)	0.76(0.01)	3333.08	2.28(0.10)	2.46(0.53)
100	0.90(0.01)	0.75(0.02)	3880.66	2.35(0.10)	4.04(0.81)
200	0.74(0.05)	0.47(0.03)	4539.58	3.73(0.47)	15.53(4.63)
300	0.60(0.06)	0.32(0.01)	6412.93	4.46(0.63)	36.71(9.81)
400	0.66(0.02)	0.35(0.04)	6375.73	4.38(0.55)	61.78(18.51)
500	0.58(0.07)	0.30(0.05)	7770.68	4.37(0.73)	86.51(31.99)

7. Conclusions and Future work

The Robust Graph mode seeking by Graph Shift [1] (RGGS) algorithm shows a great advantage in efficiently dealing with noisy data. However, no theoretical outcomes have been reported on its processing behavior and whether they converge. This leaves enormous uncertainty as to whether the RGGS algorithm works well on the test for solid theoretical reasons. In this paper, we have proposed a generic theoretical framework to prove the convergence of GS-type algorithms. A GS-type algorithm consists of three key components: simplex of generated sequence set, monotonic and continuous objective function, and closed mapping. They are mapped to the three key conditions in Zangwill's theory respectively. Consequently, the convergence of the RGGS algorithm is proved by applying Zangwill's theory.

We have shown that the framework can be applied to GS-type algorithms, as well as the Dominant set and pairwise clustering (DSPC) algorithm. Experimental results on both the RGGS algorithm and the DSPC algorithm verify that they both

converge in terms of the key aspects including the scale of the affinity matrix, the transformation number, and the iteration number.

However, as this paper limits the generated sequence set under the simplex case, more work will be done to expand it to other compact sets. In addition, Banach's contraction theory and optimization methods on non-convex mathematical programs are considered in our approach so as to make the framework more general for GS-type algorithms.

References

[1] H. Liu, S. Yan, Robust graph mode seeking by graph shift, in: ICML, 2010, pp. 671–678.

[2] J. Zhao, J. Ma, J. Tian, J. Ma, D. Zhang, A robust method for vector field learning with application to mismatch removing, in: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 2977–2984.

[3] J. Yuan, G. Zhao, Y. Fu, Z. Li, A. K. Katsaggelos, Y. Wu, Discovering thematic objects in image collections and videos., IEEE Transactions on Image Processing 21 (4) (2012) 2207–2219.

[4] T. Chen, S. Jiang, L. Chu, Q. Huang, Detection and location of near-duplicate video sub-clips by finding dense subgraphs, in: Proceedings of the 19th ACM international conference on Multimedia, ACM, 2011, pp. 1173–1176.

[5] X. Li, A. Dick, H. Wang, C. Shen, A. van den Hengel, Graph mode-based contextual kernels for robust svm tracking, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 1156–1163.

[6] X. Yang, H. Liu, L. J. Latecki, Contour-based object detection as dominant set computation, Pattern Recognition 45 (5) (2012) 1927–1936.

[7] H. Liu, S. Yan, Common visual pattern discovery via spatially coherent correspondences, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 1609–1616.

[8] M. Pavan, M. Pelillo, A new graph-theoretic approach to clustering and segmentation, in: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, Vol. 1, IEEE, 2003, pp. I–145.

[9] M. Pavan, M. Pelillo, Dominant sets and pairwise clustering, Pattern Analysis and Machine Intelligence, IEEE Transactions on 29 (1) (2007) 167–172.

[10] J. Bezdek, A convergence theorem for the fuzzy isodata clustering algorithms, Pattern Analysis and Machine Intelligence, IEEE Transactions on (1) (1980) 1–8.

[11] R. Hathaway, J. Davenport, J. Bezdek, Relational duals of the c-means clustering algorithms, Pattern recognition 22 (2) (1989) 205–212.

[12] W. Zangwill, Nonlinear programming: a unified approach, Prentice-Hall international series in management, Prentice-Hall, 1969.

[13] A. Gunawardana, W. Byrne, Convergence theorems for generalized alternating minimization procedures, The Journal of Machine Learning Research 6 (2005) 2049–2073.

[14] F. Hoppner, F. Klawonn, A contribution to convergence theory of fuzzy c-means and derivatives, Fuzzy Systems, IEEE Transactions on 11 (5) (2003) 682–694.

[15] D. Gustafson, W. Kessel, Fuzzy clustering with a fuzzy covariance matrix, in: Decision and Control including the 17th Symposium on Adaptive Processes, 1978 IEEE Conference on, Vol. 17, IEEE, 1978, pp. 761–766.

[16] V. Istratescu, Fixed point theory an introduction, Vol. 7, Kluwer Academic Print on Demand, 2002.

[17] L. Groll, J. Jakel, A new convergence proof of fuzzy c-means, Fuzzy Systems, IEEE Transactions on 13 (5) (2005) 717–720.

[18] S. Selim, M. Ismail, K-means-type algorithms: a generalized convergence theorem and characterization of local optimality, Pattern Analysis and Machine Intelligence, IEEE Transactions on (1) (1984) 81–87.

[19] H. W. Kuhn, A. W. Tucker, Nonlinear programming, in: Proceedings, Second Berkeley Symposium in Mathematical Statistics and Probability, J. Neyman, editor. University of California Pres, Berkeley, Calif, 1951, pp. 481–92.

[20] J. Weibull, Evolutionary game theory, The MIT press, 1997.

[21] J. Hofbauer, K. Sigmund, Evolutionary games and population dynamics, Cambridge University Press, 1998.

[22] R. Fisher, The genetical theory of natural selection, Clarendon Press, 1930.

[23] L. Baum, J. Eagon, An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology, Bull. Amer. Math. Soc 73 (3) (1967) 360–363.

Appendix A. Definition of $\Delta \boldsymbol{x}$

According to [1], $\Delta \boldsymbol{x} = t^* \boldsymbol{b}$, t^* and $\boldsymbol{b}$ are defined as:

$$\boldsymbol{b} = \begin{cases} -\boldsymbol{x}_i s & i \in \sigma(\boldsymbol{x}); \\ v_i, & i \notin \sigma(\boldsymbol{x}). \end{cases} \tag{A.1}$$

$$t^* = \begin{cases} \frac{1}{s}, & \text{if } \lambda s^2 + 2s\zeta - \omega \leq 0 \\ \min(\frac{1}{s}, \frac{\zeta}{\lambda s^2 + 2s\zeta - \omega}), & \text{if } \lambda s^2 + 2s\zeta - \omega > 0 \end{cases} \tag{A.2}$$

where

$$v_i = \begin{cases} 0, & i \in \sigma(\boldsymbol{x}) \\ \max(a(\boldsymbol{x}, I_i) - g(\boldsymbol{x}), 0), & i \notin \sigma(\boldsymbol{x}) \end{cases} \tag{A.3}$$

$$s = \sum_{i \notin \sigma(\boldsymbol{x})} v_i, \zeta = \sum_{i \notin \sigma(\boldsymbol{x})} v_i^2, \omega = \sum_{i,j} v_i a_{ij} v_j. \tag{A.4}$$

$g(\boldsymbol{x} + \Delta \boldsymbol{x}) - g(\boldsymbol{x}) = -(\lambda s^2 + 2s\zeta - \omega)t^2 + 2\zeta t$. When $\lambda s^2 + 2s\zeta - \omega \leq 0$, $g(\boldsymbol{x} + \Delta \boldsymbol{x}) - g(\boldsymbol{x}) \geq 0$; When $\lambda s^2 + 2s\zeta - \omega < 0$, t^* always lies in the interval

$[0, \frac{2\zeta}{\lambda s^2+2s\zeta-\omega}]$, which are the two solutions of $g(\boldsymbol{x}+\Delta\boldsymbol{x})-g(\boldsymbol{x})=0$, this leads to $g(\boldsymbol{x}+\Delta\boldsymbol{x})-g(\boldsymbol{x})>0$.

Appendix B. Proof of Proposition 1

Proof. To prove that the sequence set $\{\boldsymbol{x}_{(k)}\}_{k=0}^{\infty} \subset S$ is compact is equivalent to prove that the set is bounded and closed. Since $(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$ are all located in $[0, 1]$, the $\boldsymbol{x}$ value space is bounded. Also, from Equation (5), $\boldsymbol{e}_i = (0, ..., 1, ..., 0)$, which means the i-th component of $\boldsymbol{x}$ is 1 and the others are 0, we can denote $\boldsymbol{x}(t)$ as:

$$\boldsymbol{x}(t) = \sum_{i=1}^{n} \boldsymbol{e}_i \cdot \frac{\boldsymbol{x}(t-1)_i(A\boldsymbol{x}_i(t-1))}{\boldsymbol{x}(t-1)^T A\boldsymbol{x}(t-1)} \tag{B.1}$$

Since $\sum_{i=1}^{n} \frac{\boldsymbol{x}(t-1)_i(A\boldsymbol{x}_i(t-1))}{\boldsymbol{x}(t-1)^T A\boldsymbol{x}(t-1)} = 1$ and $0 \leq \frac{\boldsymbol{x}(t-1)_i(A\boldsymbol{x}_i(t-1))}{\boldsymbol{x}(t-1)^T A\boldsymbol{x}(t-1)} \leq 1, i = 1, \ldots, n.$. Adding $\Delta\boldsymbol{x}$ (defined in Appendix A) still holds the expression $\boldsymbol{x}(t) = \sum_{i=1}^{n} \boldsymbol{e}_i \cdot y_i, \sum_{i=1}^{n} y_i = 1$, thus $\boldsymbol{x}(t)$ is in the convex hull of S, so it is closed. Therefore, the sequence set $\{\boldsymbol{x}_{(k)}\}_{k=0}^{\infty} \subset S$ is both bounded and closed. □

Appendix C. Proof of Proposition 2

Proof. This proposition is known in mathematical biology as the fundamental theory of natural selection [21] and, in its original form, we can trace it back to [22].

We can also prove that Proposition 2 is a special case of Baum-Eagon inequality ([23]). We denote x_i as $x_i = \prod_{j=1}^{n} x_j^{\mu_{ij}}$, here $\mu_{ij} = \begin{cases} 1, & i = j; \\ 0, & i \neq j. \end{cases}$. Thus we

wish to prove that when $\boldsymbol{x}(t) \notin \Gamma$:

$$\begin{aligned} g(x) =& x^T Ax = \sum_{i=1}^{n} \omega_i x_i = \sum_{i=1}^{n} \omega_i \prod_{j=1}^{n} x_j^{\mu_{ij}} \\ &< \sum_{i=1}^{n} \omega_i \prod_{j=1}^{n} J(x_j)^{\mu_{ij}}. \end{aligned} \tag{C.1}$$

From Hölder inequation and $x_i^2 = x_i \cdot \prod_{j=1}^{n} x_j^{\mu_{ij}}$, we can get the result as:

$$\begin{aligned} g(x) &= \sum_{i=1}^{n} \{\omega_i \cdot \prod_{j=1}^{n} J(x_j)^{\mu_{ij}}\}^{\frac{1}{2}} \times \{\omega_i^{\frac{1}{2}} x_i \prod_{j=1}^{n} (\frac{1}{J(x)})^{\frac{\mu_{ij}}{2}}\} \\ &\leq \{\sum_{i=1}^{n} \omega_i \prod_{j=1}^{n} J(x_j)^{\mu_{ij}}\}^{\frac{1}{2}} \times \{\sum_{i=1}^{n} \omega_i x_i \prod_{j=1}^{n} (\frac{x_j}{J(x_j)})^{\mu_{ij}}\}^{\frac{1}{2}} \end{aligned} \tag{C.2}$$

Equality holds if and only if $\forall p, q \in \{1, \cdots, n\}$,

$$\begin{aligned} &\frac{\{\omega_p \cdot \prod_{j=1}^{n} J(x_j)^{\mu_{pj}}\}^{\frac{1}{2}}}{\omega_p^{\frac{1}{2}} x_p \prod_{j=1}^{n} (\frac{1}{J(x)})^{\frac{\mu_{pj}}{2}}} = \frac{\{\omega_q \cdot \prod_{j=1}^{n} J(x_j)^{\mu_{qj}}\}^{\frac{1}{2}}}{\omega_q^{\frac{1}{2}} x_q \prod_{j=1}^{n} (\frac{1}{J(x)})^{\frac{\mu_{qj}}{2}}} \\ \Longleftrightarrow\ &\frac{J(x_p)}{x_p} = \frac{J(x_q)}{x_q} \iff \omega_p = \omega_q \end{aligned} \tag{C.3}$$

Using the inequality of geometric and arithmetic means to the double products of

the second brace, we can conclude:

$$\begin{aligned}
&\sum_{i=1}^{n}\omega_i x_i \prod_{j=1}^{n}(\frac{x_j}{J(x_j)})^{\mu_{ij}} \leq \sum_{i=1}^{n}\omega_i x_i \sum_{j=1}^{n}\mu_{ij}\cdot\frac{x_j}{J(x_j)} \\
&= \sum_{i=1}^{n}\omega_i x_i \sum_{j=1}^{n}\mu_{ij}x_j \cdot \frac{\sum_{k=1}^{n}\omega_k x_k}{\omega_j x_j} \\
&= \sum_{k=1}^{n}\omega_k x_k \cdot \sum_{j=1}^{n} x_j \cdot \frac{\sum_{i=1}^{n}\omega_i x_i \cdot \mu_{ij}}{\omega_j x_j} \\
&= \sum_{k=1}^{n}\omega_k x_k \cdot \sum_{j=1}^{n} x_j = \sum_{k=1}^{n}\omega_k x_k.
\end{aligned} \tag{C.4}$$

Equality holds if and only if $\forall p, q \in \{1, \cdots, n\}$,

$$\frac{x_p}{J(x_p)} = \frac{x_p}{J(x_p)} \iff \omega_p = \omega_q. \tag{C.5}$$

Here the last equation succeed because $\omega_i x_i \cdot \mu_{ij} = \omega_j x_j$ if and only if $j = i$, otherwise it is 0, and $\sum_{j=1}^{n} x_j = 1$.

we put the result into the second braces, and get

$$\begin{aligned}
&\sum_{i=1}^{n}\omega_i x_i \leq \{\sum_{i=1}^{n}\omega_i \prod_{j=1}^{n} J(x_j)^{\mu_{ij}}\}^{\frac{1}{3}} \times \{\sum_{i=1}^{n}\omega_i x_i\}^{\frac{2}{3}} \\
\iff &\{\sum_{i=1}^{n}\omega_i x_i\}^{\frac{1}{3}} \leq \{\sum_{i=1}^{n}\omega_i \prod_{j=1}^{n} J(x_j)^{\mu_{ij}}\}^{\frac{1}{3}} \\
\iff &\sum_{i=1}^{n}\omega_i x_i \leq \sum_{i=1}^{n}\omega_i \prod_{j=1}^{n} J(x_j)^{\mu_{ij}}
\end{aligned} \tag{C.6}$$

Equality holds if and only if $\omega_i = \omega_j, \forall i, j \in \{1, \cdots, n\}$. It is the situation contained by the solution set Γ.

Thus, the function $g(x) = \boldsymbol{x}^t A \boldsymbol{x}$ is strictly increasing along any nonconstant trajectory of (5) when $\boldsymbol{x} \in X/\Gamma$. □

Appendix D. Proof of Proposition 4

Proof. The continuity of $g(\boldsymbol{x}) = \boldsymbol{x}^T A \boldsymbol{x}$ is obvious. We here discuss the increasing monotonicity. From Equation (4), we have

$$g(\boldsymbol{x}) \leq g(C(\boldsymbol{x})) < g(B \circ C(\boldsymbol{x})) = g(T_m(\boldsymbol{x})) \tag{D.1}$$

□

Appendix E. Proof of Proposition 5

Proof. If $\boldsymbol{x}^0 \in X$, then $\boldsymbol{x}^n \to \boldsymbol{x}^0$ and $\boldsymbol{y}^n \to \boldsymbol{y}^0$ when $n \to \infty$. Consequently $\boldsymbol{y}^n \in C(\boldsymbol{x}^n)$, indicating that

$$\boldsymbol{y}^n = \boldsymbol{x}^n + \Delta \boldsymbol{x}^n \tag{E.1}$$

and we need to prove that

$$\boldsymbol{y}^0 = C(\boldsymbol{x}^0) = \boldsymbol{x}^0 + \Delta \boldsymbol{x}^0. \tag{E.2}$$

This lies in two situations:

(1) $\boldsymbol{x}^0$ is in Γ, thus $\Delta\boldsymbol{x}^0 = 0$.

(2) $\boldsymbol{x}^0$ is in $X\backslash\Gamma$.

For situation (2), as $\boldsymbol{x}^n \to \boldsymbol{x}^0$ and $\boldsymbol{y}^n \to \boldsymbol{y}^0$, we can find a large N such that $\forall\varepsilon > 0, \exists n > N,$ so that $|\boldsymbol{x}^n - \boldsymbol{x}^0| < \varepsilon, |\boldsymbol{y}^n - \boldsymbol{y}^0| < \varepsilon$. Consequently, we have

$$\begin{aligned} &|\boldsymbol{y}^0 - C(\boldsymbol{x}^0)| \le |\boldsymbol{y}^0 - \boldsymbol{y}^n| + |\boldsymbol{y}^n - C(\boldsymbol{x}^n)| + |C(\boldsymbol{x}^n) \\ &- C(\boldsymbol{x}^0)| \le \varepsilon + |\boldsymbol{x}^n - \boldsymbol{x}^0| + |\Delta\boldsymbol{x}^0 - \Delta\boldsymbol{x}^n| \\ &\le 2\varepsilon + |\Delta\boldsymbol{x}^0 - \Delta\boldsymbol{x}^n| \end{aligned} \tag{E.3}$$

According to Lemma 2 and the definitions of $\boldsymbol{b}$ and t, it is a continuous mapping on $\boldsymbol{x}$, this results in that $\forall\varepsilon > 0, \exists\delta$ such that $|\Delta\boldsymbol{x}^0 - \Delta\boldsymbol{x}^n| \le \varepsilon$. So, $\forall\varepsilon > 0$, if N_1 is big enough, $n > N_1, |\boldsymbol{x}^n - \boldsymbol{x}^0| <min(\varepsilon/3, \delta)$, $|\boldsymbol{y}^n - \boldsymbol{y}^0| < \varepsilon/3$, then

$$|\boldsymbol{y}^0 - C(\boldsymbol{x}^0)| \le \varepsilon \tag{E.4}$$

Hence $\boldsymbol{y}^0 = C(\boldsymbol{x}^0)$, and C is closed on $X\backslash\Gamma$. □