Mapping Market Structure Evolution

Maximilian Matthe (Goethe University Frankfurt, Germany)
matthe@wiwi.uni-frankfurt.de
Daniel M. Ringel (University of North Carolina at Chapel Hill, USA)
dmr@kenan-flagler.unc.edu
Bernd Skiera (Goethe University Frankfurt, Germany)
skiera@wiwi.uni-frankfurt.de

ONLINE APPENDIX

A. Gradient Derivation and Implementation

Herein, we formally derive the gradient of EvoMap's cost function to facilitate the implementation of our framework for different methods, optimization procedures, and various programming languages. We estimate the sequence of maps $(\hat{X}_t)_{t=1,\dots,T}, \hat{X}_t \in \mathbb{R}^{n\times 2}$ by optimizing the following cost function:

$$C_{total}(X_1,\dots,X_T) = \sum_{t=1}^{T} C_{static}(X_t) + \alpha \cdot C_{temporal}(X_1,\dots,X_T) \tag{A-1}$$

Therefore, the gradient $\text{grad}\, C_{total}(X_1,\dots,X_T)$ consists of the two independent components $\text{grad}\, C_{static}(X_t)$ and $\text{grad}\, C_{temporal}(X_1,\dots,X_T)$. The chosen mapping method provides $\text{grad}\, C_{static}(X_t)$. We derive $\text{grad}\, C_{temporal}(X_1,\dots,X_T)$ subsequently. Recall that we defined the temporal cost function as

$$C_{temporal}(X_1,\dots,X_T) = \sum_{i=1}^{n} f_w(i) \sum_{k=1}^{p} \sum_{t=k+1}^{T} 1_{[i\in I_{t,k}]} \left\| \nabla^k x_{i,t} \right\|^2 \tag{A-2}$$

Fix a period $\tau \in \{1,\dots,T\}$ and a firm $j \in \{1,\dots,n\}$ and let $x_{j,\tau} \in \mathbb{R}^2$ denote firm j's map position at time τ. $\text{grad}\, C_{temporal}$ consists of all partial derivatives of $C_{temporal}$ w.r.t. $x_{j,\tau}\ \forall j \in \{1,\dots,n\}, \tau \in \{1,\dots T\}$, expressed by

$$\frac{\partial C_{temporal}}{\partial x_{j,\tau}} = f_w(j) \sum_{k=1}^{p} \sum_{t=k+1}^{T} 1_{[j\in I_{t,k}]} \frac{\partial \left\| \nabla^k x_{j,t} \right\|^2}{\partial x_{j,\tau}} \tag{A-3}$$

This partial derivative of the temporal component for firm *j* does not depend on any other firm's position. Suppose that $j \in I_{t,k}$ (else, the respective term in the sum is zero). $f_w(j)$ only depends upon the input data and is thus a scalar independent of $x_{j,\tau}$. The derivation of the partial derivative of $C_{temporal}$ therefore consists of deriving the partial derivative of $\left\|\nabla^k x_{j,t}\right\|^2$ w.r.t $x_{j,\tau}$ for all $k \in \{1, \dots, p\}$ and $t \in \{1, \dots, T\}$. The partial derivative of the outer norm depends on its choice. We implement EvoMap for the Euclidean norm, such that $\frac{\partial \|\mathbf{v}\|^2}{\partial \mathbf{v}} = 2\mathbf{v}$ for any real-valued vector $\mathbf{v}$. Therefore,

$$\frac{\partial \left\|\nabla^k x_{j,t}\right\|^2}{\partial x_{j,\tau}} = 2 \cdot \nabla^k x_{j,t} \cdot \frac{\partial \nabla^k x_{j,t}}{\partial x_{j,\tau}} \tag{A-4}$$

Recall that the inner part of the norm corresponds to the *k*-th order difference of firm *j*'s map position at time t, formally defined as

$$\nabla^k x_{j,t} := \begin{cases} \left(\nabla^{k-1} x_{j,t} - \nabla^{k-1} x_{j,t-1}\right) & \text{for } k \geq 1 \wedge t \geq k+1 \\ x_{j,t} & k = 0 \\ 0 & else \end{cases} \tag{A-5}$$

Thus, we can derive the partial derivative of the inner part of the norm, $\frac{\partial \nabla^k x_{j,t}}{\partial x_{j,\tau}}$ as follows. Assume that $k \geq 1 \wedge t \geq k+1$ (else, the partial evaluates to 1, if $k = 0$, or 0). From the definition of $\nabla^k x_{j,t}$ in (A-5), it follows that:

$$\frac{\partial \nabla^k x_{j,t}}{\partial x_{j,\tau}} = \left(\frac{\partial \nabla^{k-1} x_{j,t}}{\partial x_{j,\tau}} - \frac{\partial \nabla^{k-1} x_{j,t-1}}{\partial x_{j,\tau}}\right) \tag{A-6}$$

such that we can derive $\frac{\partial \nabla^k x_{j,t}}{\partial x_{j,\tau}}$ recursively, starting with $k = 1$:

$$\frac{\partial \nabla^1 x_{j,t}}{\partial x_{j,\tau}} = \left(\frac{\partial \nabla^0 x_{j,t}}{\partial x_{j,\tau}} - \frac{\partial \nabla^0 x_{j,t-1}}{\partial x_{j,\tau}}\right) \tag{A-7}$$

where

$$\frac{\partial \nabla^0 x_{j,t}}{\partial x_{j,\tau}} = \frac{\partial x_{j,t}}{\partial x_{j,\tau}} = \begin{cases} 1 \; if \; t = \tau \\ 0 \; else \end{cases} \tag{A-8}$$

and

$$\frac{\partial \nabla^0 x_{j,t-1}}{\partial x_{j,\tau}} = \frac{\partial x_{j,t-1}}{\partial x_{j,\tau}} = \begin{cases} 1 \; if \; t = \tau + 1 \\ 0 \; else \end{cases} \tag{A-9}$$

We derive the second term in (A-6) from the first term by shifting the time indices by one period:

$$\frac{\partial \nabla^k x_{j,t-1}}{\partial x_{j,\tau}} = \frac{\partial \nabla^k x_{j,t}}{\partial x_{j,\tau+1}} \tag{A-10}$$

By inserting (A-8) and (A-9), we can then express (A-7) as

$$\frac{\partial \nabla^1 x_{j,t}}{\partial x_{j,\tau}} = \begin{cases} 1, \text{if t} = \tau \\ -1, \text{if t} = \tau + 1 \\ 0 \text{ else} \end{cases} \tag{A-11}$$

which yields the partial $\frac{\partial \nabla^k x_{j,t}}{\partial x_{j,\tau}}$ for all *t* and $k = 1$. From that, we derive the expressions for $k = 2$ analogously to (A-7):

$$\frac{\partial \nabla^2 x_{j,t}}{\partial x_{j,\tau}} = \left(\frac{\partial \nabla^1 x_{j,t}}{\partial x_{j,\tau}} - \frac{\partial \nabla^1 x_{j,t-1}}{\partial x_{Ij\tau}} \right) \tag{A-12}$$

Where we have already derived the first term in (A-11) and we can obtain the second term from (A-11) after shifting time indices by one period:

$$\frac{\partial \nabla^2 x_{j,t-1}}{\partial x_{j,\tau}} = \begin{cases} 1, if \; t = \tau + 1 \\ -1, if \; t = \tau + 2 \\ 0 \; else \end{cases} \tag{A-13}$$

Such that the partial $\frac{\partial \nabla^k x_{j,t}}{\partial x_{j,\tau}}$ for all *t* and $k = 2$ can be expressed as

$$\frac{\partial \nabla^2 x_{j,t}}{\partial x_{j,\tau}} = \begin{cases} 1, if \; t = \tau \\ -2, if \; t = \tau + 1 \\ 1, if \; t = \tau + 2 \\ 0, else \end{cases} \tag{A-14}$$

We can derive the remaining partials for *k* higher than two similarly by repeating the steps between (A-12) and (A-14). We obtain the final gradient by inserting all results into (A-4) and (A-3).

Based on these derivations, researchers can implement EvoMap for a given static mapping method as follows. We assume that the chosen method has a non-negative cost function C_{static} (i.e., lower values correspond to better solutions) and can be optimized iteratively via gradient-based methods. Provided that method, one needs to proceed as follows:

First, adopt (or derive) the static gradient $\text{grad}\, C_{static}$ entailing the partial derivatives $\begin{bmatrix} \frac{\partial C_{static}}{\partial x_{1,\tau}} \\ \vdots \\ \frac{\partial C_{static}}{\partial x_{n,\tau}} \end{bmatrix}$ for any map layout $X_\tau \in (X_t)_{t=1,\dots,T}$. If any firm is not present at time τ, set its respective entry in the gradient to zero. Then, combine these gradients for all periods: $\text{grad} \sum_{\tau=1}^{T} C_{static} = \begin{bmatrix} \text{grad}\, C_{static}(X_1) \\ \vdots \\ \text{grad}\, C_{static}(X_T) \end{bmatrix}$. Here, we simply stacked all map layouts in a temporal order.

Second, derive the partial derivatives $\begin{bmatrix} \frac{\partial C_{temp}}{\partial x_{1,\tau}} \\ \vdots \\ \frac{\partial C_{temp}}{\partial x_{n,\tau}} \end{bmatrix}$ of $C_{temporal}$ with respect to X_τ for any map layout $X_\tau \in (X_t)_{t=1,\dots,T}$ using (A-3) and the expressions that follow it. Analogously to the static gradients, stack them in temporal order to obtain the final temporal gradient $\text{grad}\, C_{temporal}$.

Finally, combine the two (stacked) gradients according to (A-1). One can then use iterative optimization techniques to find the cost-minimizing sequence of map layouts (e.g., using the same optimization routine commonly used for the given static mapping method).

B. Numerical Examples for Penalty Derivations

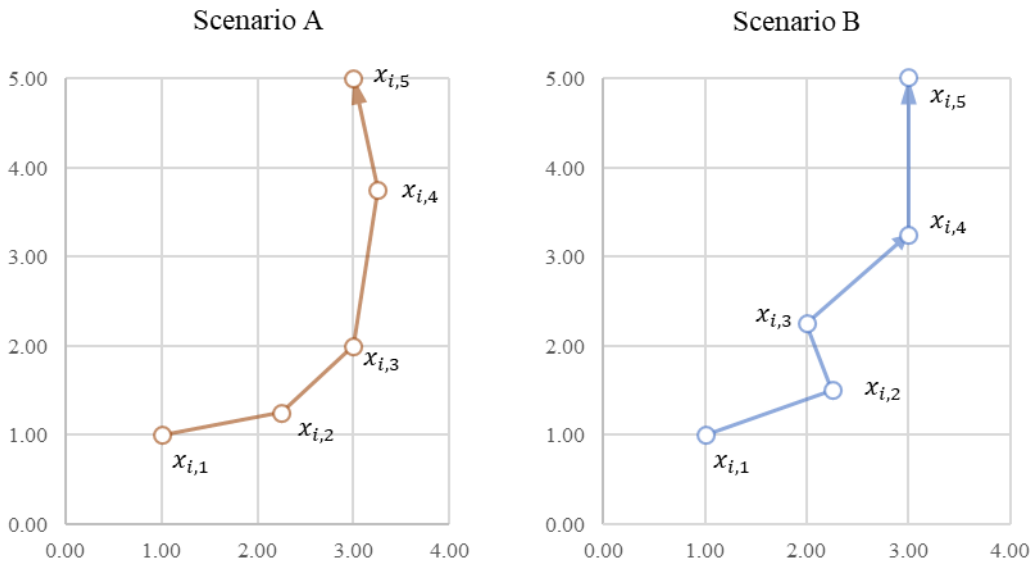
Herein, we provide a numerical example for the derivation of the penalty terms in EvoMap's cost function. Specifically, we consider the same firm *i* under two hypothetical scenarios:

- Scenario A: Firm *i* moves gradually (left graph in Online Appendix Figure B-1)
- Scenario B: Firm *i* moves more erratically (right graph in Online Appendix Figure B-1)

We use these hypothetical trajectories to demonstrate how our cost function penalizes the solution in scenario B stronger than in scenario A (assuming everything else is equal). For both scenarios, Online Appendix Table B-1 displays the positions $x_{i,t}$, the resultant k-th order differences $\nabla^k x_{i,t}$ for $k \in \{1,2\}$, and the corresponding values of their norm $\|\nabla^k x_{i,t}\|^2$ which enter the cost function. The numerical example demonstrates that the cost function will take higher values (≈ 13.00 vs. ≈ 9.75) under scenario B. Thus, all else being equal (for instance, the static cost function values), the cost function favors the solution in scenario A. The example also demonstrates the added value of incorporating higher-order differences: When considering

only $\|\nabla^k x_{i,t}\|^2$ for $k = 1$ both trajectories yield equivalent cost function values of 7.50.

Online Appendix Figure B-1: Two Different Trajectories



Notes: Axes correspond to the two map dimensions.

Online Appendix Table B-1: Numerical Example for Penalty Derivations

	Scenario A (gradual trajectory)					Scenario B (more erratic trajectory)				
Symbolic Expression	$x_{i,t}$ (Positions)	$\nabla^1 x_{i,t}$ (1st-Order Differences)	$\left\|\nabla^1 x_{i,t}\right\|^2$ (Sq. Eucl. Norm)	$\nabla^2 x_{i,t}$ (2nd-Order Differences)	$\left\|\nabla^2 x_{i,t}\right\|^2$ (Sq. Eucl. Norm)	$x_{i,t}$ (Positions)	$\nabla^1 x_{i,t}$ (1st-Order Differences)	$\left\|\nabla^1 x_{i,t}\right\|^2$ (Sq. Eucl. Norm)	$\nabla^2 x_{i,t}$ (2nd-Order Differences)	$\left\|\nabla^2 x_{i,t}\right\|^2$ (Sq. Eucl. Norm)
Time t = 1	$\begin{bmatrix}1.00\\1.00\end{bmatrix}$	--	--	--	--	$\begin{bmatrix}1.00\\1.00\end{bmatrix}$	--	--	--	--
Time t = 2	$\begin{bmatrix}2.25\\1.25\end{bmatrix}$	$\begin{bmatrix}1.25\\0.25\end{bmatrix}$	≈ 1.63	--	--	$\begin{bmatrix}2.25\\1.50\end{bmatrix}$	$\begin{bmatrix}1.25\\0.50\end{bmatrix}$	≈ 1.81	--	--
Time t = 3	$\begin{bmatrix}3.00\\2.00\end{bmatrix}$	$\begin{bmatrix}0.75\\0.75\end{bmatrix}$	≈ 1.13	$\begin{bmatrix}-0.50\\0.50\end{bmatrix}$	$= 0.50$	$\begin{bmatrix}2.00\\2.25\end{bmatrix}$	$\begin{bmatrix}-0.25\\0.75\end{bmatrix}$	≈ 0.63	$\begin{bmatrix}-1.50\\0.25\end{bmatrix}$	≈ 2.31
Time t = 4	$\begin{bmatrix}3.25\\3.75\end{bmatrix}$	$\begin{bmatrix}0.25\\1.75\end{bmatrix}$	≈ 3.13	$\begin{bmatrix}-0.50\\1.00\end{bmatrix}$	$= 1.25$	$\begin{bmatrix}3.00\\3.25\end{bmatrix}$	$\begin{bmatrix}1.00\\1.00\end{bmatrix}$	$= 2.00$	$\begin{bmatrix}1.25\\0.25\end{bmatrix}$	≈ 1.63
Time t = 5	$\begin{bmatrix}3.00\\5.00\end{bmatrix}$	$\begin{bmatrix}-0.25\\1.25\end{bmatrix}$	≈ 1.63	$\begin{bmatrix}-0.50\\-0.50\end{bmatrix}$	$= 0.50$	$\begin{bmatrix}3.00\\5.00\end{bmatrix}$	$\begin{bmatrix}0.00\\1.75\end{bmatrix}$	≈ 3.06	$\begin{bmatrix}-1.00\\0.75\end{bmatrix}$	≈ 1.56
$\sum_t \left\|\nabla^k x_{i,t}\right\|$	--	--	$= 7.50$	--	$= 2.25$	--	--	$= 7.50$	--	$= 5.50$
$\sum_t \sum_k \left\|\nabla^k x_{i,t}\right\|$	--	$= \mathbf{9.75}$				--	$= \mathbf{13.0}$			

Notes: $\|\cdot\|^2$ denotes the Squared Euclidean Norm (Sq. Eucl. Norm).

C. Additional Simulation Results

In what follows, we extend our simulation study to (1) investigate the benefits of EvoMap's adaptive regularization and smoothing properties, (2) test alternative specifications of two dynamic mapping metrics, and (3) improve Dynamic t-SNE's volatile performance by using EvoMap's optimization procedure.

Specifically, we introduce the following two variants of EvoMap to our simulation study: EvoMap (adaptive regularization only), which introduces firm-specific weights but ignores higher-order differences, and EvoMap (smoothing only), which considers higher-order differences but does not include any firm-specific weights. Considering the goodness-of-fit measures (HIT-RATE and C-CORR), we find that adaptive regularization increases dynamic goodness-of-fit (C-CORR increases), while static goodness-of-fit remains similar (HIT-RATE does not change). Thereby, the descriptive metrics reveal that the resultant maps are slightly less aligned, and trajectories become slightly less gradual (PERS decreases).

These findings show that adaptive regularization increases flexibility and allows EvoMap to recover actual market structure changes better. In contrast, smoothing results in more aligned positions (MIS-ALIGN decreases) and more gradual trajectories (PERS increases). Moreover, it also increases goodness-of-fit (recall that we evaluate goodness-of-fit against the simulated positions before adding any noise). Combining smoothing with adaptive regularization yields the highest goodness-of-fit of all alternatives.

Next, we introduce two additional metrics to our simulation study to demonstrate that our results are robust to alternative specifications of the evaluation metrics. Specifically, we propose the following alternative metrics:

- ALIGN (Alignment): An alternative measure to misalignment, using the average cosine similarity of successive positions across all periods and firms. Alignment varies between -1 and 1, where positive values indicate high alignment (and vice versa).
- CPA (Change Prediction Accuracy): An alternative measure to change correlation. Rather than simply using the correlation of simulated vs. recovered movement path lengths, we estimate a logistic regression model with a binary dependent variable "change" (1 if a firm was affected by one of the evolution scenarios within the simulation, 0 else) and a single independent variable "trajectory length" (total length of the movement path on the estimated market structure map). We evaluate its predictive accuracy. Since our sample is imbalanced with a large fraction of static and

a small fraction of dynamic positions, we use the F_1 score based on precision and recall as a measure of predictive accuracy.

Consistent with MIS-ALIGN and C-CORR, EvoMap also outperforms extant dynamic mapping approaches on ALIGN (Alignment) and CPA (Change Prediction Accuracy). Naturally, as the ALIGN metric is bounded between -1 and 1, it varies less than the unbounded MIS-ALIGN metric across different methods. See Online Appendix Table C-1 for details.

Finally, we improve the relatively poor performance of Dynamic t-SNE (Rauber et al. 2016), which they provide at https://github.com/paulorauber/thesne (latest version as of 03/22/2016), by using EvoMap's optimization procedure. Specifically, we run EvoMap without its adaptive regularization and smoothing components (such that its cost function equals Dynamic t-SNE's cost function up to a constant). Our optimization procedure improves all mapping quality metrics (see Online Appendix Table C-1). However, these improvements still fall short of EvoMap's mapping quality.

Online Appendix Table C-1: Simulation Results with Additional Evaluation Metrics

Mapping Approach	ALIGN	MIS-ALIGN	PERS	C-CORR	CPA	HIT-RATE
(I) EvoMap (t-SNE)	0.997 (0.004)	0.155 (0.079)	0.875 (0.079)	0.835 (0.145)	0.750 (0.288)	0.665 (0.105)
(II) EvoMap (t-SNE) - smoothing only	0.998 (0.004)	0.146 (0.076)	0.888 (0.071)	0.817 (0.147)	0.737 (0.289)	0.665 (0.105)
(III) EvoMap (t-SNE) - adaptive regularization only	0.993 (0.019)	0.240 (0.167)	0.512 (0.249)	0.716 (0.266)	0.656 (0.343)	0.658 (0.109)
(IV) Dynamic t-SNE - via EvoMap	0.994 (0.018)	0.224 (0.166)	0.539 (0.247)	0.693 (0.266)	0.637 (0.338)	0.659 (0.109)
(V) Dynamic t-SNE - Rauber et al. (2016)	0.935 (0.100)	20.636 (33.605)	0.370 (0.310)	0.291 (0.299)	0.228 (0.361)	0.401 (0.152)
(VI) Ex-post Alignment (t-SNE)	0.728 (0.291)	12.238 (14.496)	-0.379 (0.191)	0.129 (0.305)	0.069 (0.218)	0.642 (0.120)
(VII) Sequential Initialization (t-SNE)	0.965 (0.043)	2.474 (1.857)	0.044 (0.275)	0.287 (0.292)	0.214 (0.287)	0.634 (0.123)
(VIII) Fixed Initialization (t-SNE)	0.832 (0.199)	4.808 (3.992)	-0.317 (0.186)	0.078 (0.247)	0.064 (0.194)	0.643 (0.120)
(IX) Independent Mapping (t-SNE)	0.003 (0.125)	21.920 (16.093)	-0.484 (0.098)	-0.098 (0.215)	0.018 (0.119)	0.642 (0.120)

Notes: Reported metrics are averages over 2,187 simulation iterations corresponding to the parameter space reported in Appendix A. Standard deviation in parentheses. Dynamic t-SNE (Rauber et al. 2016): Dynamic t-SNE, as provided by its authors on GitHub. Dynamic t-SNE (via EvoMap): Dynamic t-SNE optimized via EvoMap's optimization procedure. We set hyperparameters as described in Appendix A. Their distributions are similar to the ones reported in Appendix A. For Dynamic t-SNE (Rauber et al. 2016), we set its remaining parameters to the default values provided by its authors.

D. Sensitivity Analysis

We investigate the sensitivity of EvoMap's dynamic mapping quality to the various market structure characteristics. To do so, we regress the evaluation criteria on the simulation parameters listed in Appendix Table A-1 (using linear regression with dummy-coded independent variables). We exclude one level of each simulation parameter in each regression model as its reference point.

Online Appendix Table D-1 reports the results for each of the four estimated regression models (1) to (4).

Online Appendix Table D-1: Regression Results for Relationships between EvoMap's Dynamic Mapping Quality and Simulation Parameters

Market structure characteristic	Simulation parameter	Value	(1) MIS-ALIGN	(2) C-CORR	(3) PERS	(3) HIT-RATE
		Constant	0.0965*** (0.0045)	0.7719*** (0.0108)	0.8195*** (0.0058)	0.6417*** (0.0059)
Evolution	Scenario	II (Shifts in positions)	0.0505*** (0.0029)	0.1075*** (0.0068)	0.0028 (0.0036)	-0.0082** (0.0037)
		III (Market entry)	0.0272*** (0.0029)	0.0792*** (0.0068)	0.0149*** (0.0036)	-0.0168*** (0.0037)
Number of firms	n	100	0.0449*** (0.0029)	0.0398*** (0.0068)	0.0341*** (0.0036)	0.0602*** (0.0037)
		250	0.0681*** (0.0029)	0.0336*** (0.0068)	0.0629*** (0.0036)	-0.0705*** (0.0037)
Number of dimensions	d	8	-0.014*** (0.0029)	0.0146** (0.0068)	0.0175*** (0.0036)	-0.0163*** (0.0037)
		16	-0.0271*** (0.0029)	0.0357*** (0.0068)	0.033*** (0.0036)	-0.0296*** (0.0037)
Number of submarkets	k	8	-0.0527*** (0.0029)	-0.007 (0.0068)	0.0133*** (0.0036)	0.0989*** (0.0037)
		12	-0.071*** (0.0029)	-0.0283*** (0.0068)	0.0037 (0.0036)	0.1177*** (0.0037)
Within submarket standard deviation	σ_{sub}	0.075	-0.0056* (0.0029)	-0.0114* (0.0068)	-0.0006 (0.0036)	0.0092** (0.0037)
		0.150	-0.0142*** (0.0029)	-0.0298*** (0.0068)	-0.0026 (0.0036)	-0.0116*** (0.0037)
Temporal noise	$\sigma_{temporal}$	0.025	0.0245*** (0.0029)	-0.026*** (0.0068)	-0.0168*** (0.0036)	-0.0104*** (0.0037)
		0.050	0.0582*** (0.0029)	-0.0769*** (0.0068)	-0.0441*** (0.0036)	-0.0413*** (0.0037)
Share of firms affected	ρ	0.100	0.0331*** (0.0029)	0.0227*** (0.0068)	0.018*** (0.0036)	-0.0043 (0.0037)
		0.150	0.0524*** (0.0029)	0.0364*** (0.0068)	0.0296*** (0.0036)	-0.0066* (0.0037)
Observations			2,187	2,187	2,187	2,187
R-squared			0.53	0.20	0.23	0.55

Standard errors in parentheses
*** p<0.01, ** p<0.05, * p <0.10., std.: standard deviation

E. Further Implementation Details and Simulation Runtime Estimates

For this paper, we implement EvoMap using Python version 3.7.2. We use the following additional packages:

Online Appendix Table E-1: Core Implementation Details

Package Name	Version Used	Use-Case
numpy	1.21.2	Numerical calculations
scipy	1.7.1	Efficient calculation of Euclidean distances and norms
numba	0.53.0	Just-in-time compilation for speed optimization

Our results (not shown here) show that EvoMap's runtime increases approximately quadratically with the number of firms and linearly with the number of periods[1]. Thus, running EvoMap once takes roughly the same time as running an existing mapping method independently for each period (assuming both implementations are equally efficient in their computations). The only difference is the more complex calculation of the gradient. Online Appendix Table E-2 presents the average runtime for all methods used in our simulations and shows that EvoMap's more complex gradient does not substantially affect runtime in our simulations.

Online Appendix Table E-2: Runtime Comparison across Methods

Method	Average Runtime in Seconds
EvoMap (t-SNE)	140.17 (159.47)
Dynamic t-SNE (via EvoMap)	128.79 (156.42)
Dynamic t-SNE (Rauber et al. 2016)	250.32 (234.39)
Independent t-SNE	130.88 (157.04)

Notes: Averages across 2,187 simulation iterations. Standard deviation in parentheses.

F. Extended TNIC Analysis using EvoMap with MDS

The main article presents EvoMap's dynamic market structure map for the full sample of 1,092 firms. For our analysis of the full sample, we paired EvoMap with t-SNE. Here, we aim to extend this analysis using EvoMap paired with an alternative mapping method (specifically, metric MDS). Our objective is twofold: First, to

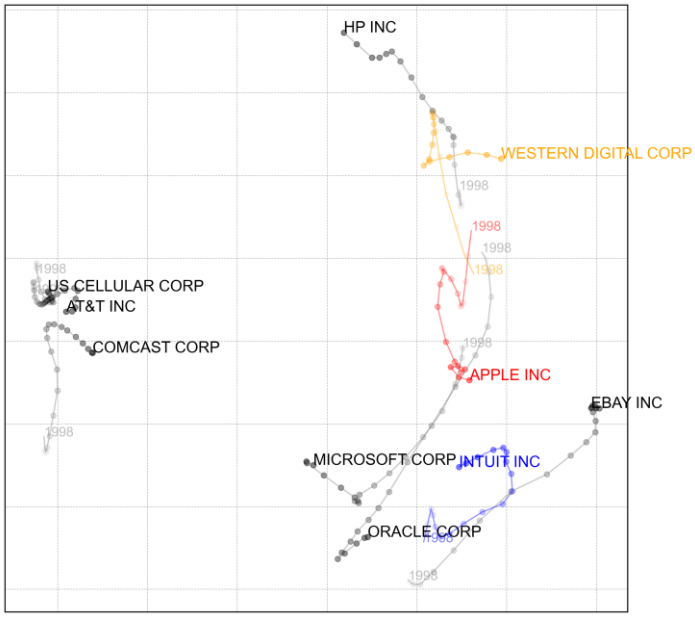
[1] Our implementation of EvoMap, used throughout our simulations, uses the following default parameters: For t-SNE, we set perplexity to 15, the initial learning rate to 20, initial momentum factor to .5, final momentum factor to .8, the switch iteration to 250, the early exaggeration factor to 4, and the maximum number of iterations to 2000. For the Stress-based alternatives (MDS and Sammon), we set the initial learning rate to 0.1 and the maximum number of iterations to 2000. We automatically adjust the learning rate (via a factor of 0.1) for all methods in case the gradient diverges during optimization.

demonstrate the benefits of EvoMap's ability to easily accommodate alternative static mapping methods for practical competitive analysis applications. And second, to test the robustness of our empirical findings towards the choice of mapping method. To do so, we generate a second dynamic market structure map using EvoMap paired with metric MDS. As metric MDS, however, does not scale well to many firms, we focus this analysis on a smaller sample of firms from the technology sector. We then test if the resultant (small) map for EvoMap paired with metric MDS resembles the same patterns as the corresponding area in the (large) map for EvoMap paired with t-SNE.

We motivate the analysis of market structure evolution with two different static mapping methods (i.e., t-SNE and metric MDS) as follows: Imagine a market analyst required a birds-eye view on how a market comprised of many firms and submarkets evolved. Because mapping quality tends to be superior with t-SNE in such large *N* settings, the analyst chooses to first pair EvoMap with t-SNE (similar to the analysis in the main article). Having identified some high-level trends and interesting firm trajectories, the analyst now wants to study a specific area of the dynamic market structure map more thoroughly (e.g., a specific submarket, a sector, or the set of a firm's imminent competitors). Because metric MDS provides some advantages in smaller *N* settings over t-SNE (e.g., better interpretability of the resultant map distances), the analyst pairs EvoMap with metric MDS to generate a second dynamic market structure map for the subsample of firms in the area of interest.

We replicate the above-outlined scenario for our empirical application as follows. We first select a small, technology-focused subsample of all firms. Our subsample includes the following 10 firms from the technology sector: AT&T, US Cellular, Comcast, Microsoft, Western Digital, HP, Oracle, eBay, Intuit, Apple. We then estimate a dynamic market structure map for this subsample using EvoMap paired with metric MDS (we set $\alpha = 9.1 \times 10^{-1}$ and $p = 2$).

Online Appendix Figure F-1: Dynamic Market Structure Map for 10 Technology Firms 1998 – 2017 (estimated using EvoMap paired with metric MDS)



Notes: Trajectories correspond to 20 successive years between 1998 and 2017 (firm name labels last period).

While the smaller map reveals additional nuances, the overall insights do not change much. For example, Apple's trajectory diverges from Western Digital and converges with Intuit (as is the case in the larger dynamic market structure map in Figure 7). As such, our findings remain robust across two different samples and two different static mapping methods used with EvoMap.