Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Chair for
INFORMATICS I
Efficient Algorithms and
Knowledge-Based Systems

Institute for Informatics

Using the A* Algorithm to Find Optimal Sequences for Area Aggregation

Dongliang Peng[1], Alexander Wolff[1], Jan-Henrik Haunert[2]

[1]Chair of Computer Science I, University of Würzburg, Germany
[2]Institute of Geodesy and Geoinformation, University of Bonn, Germany

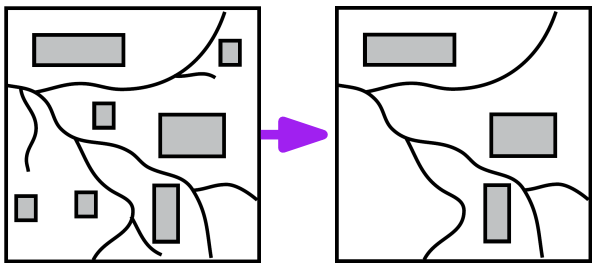
Map Generalization...

...is about selecting and representing information on a map.

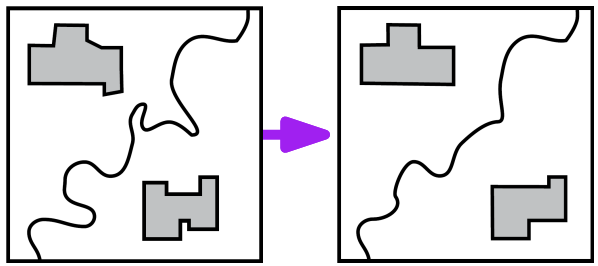
Map Generalization...

...is about selecting and representing information on a map.

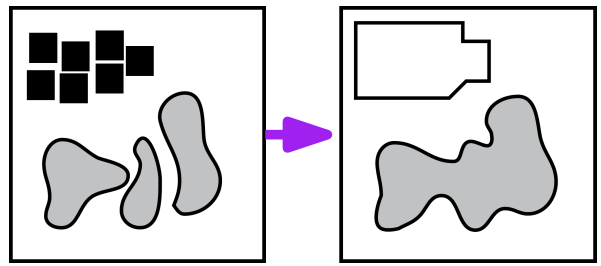
Typical generalization operators (ESRI 1996):



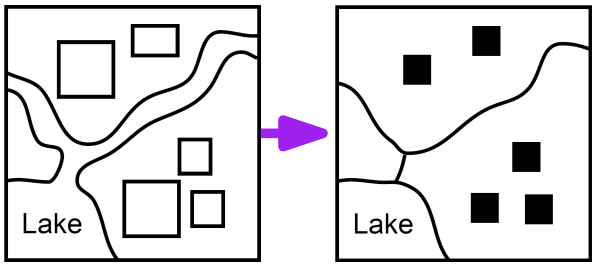
Elimination



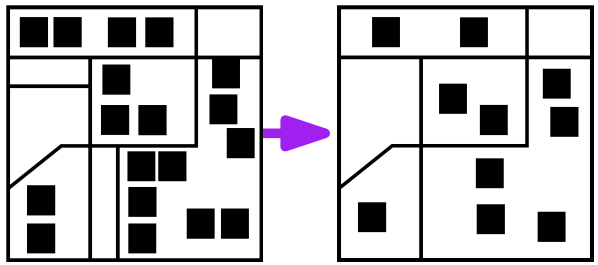
Simplification



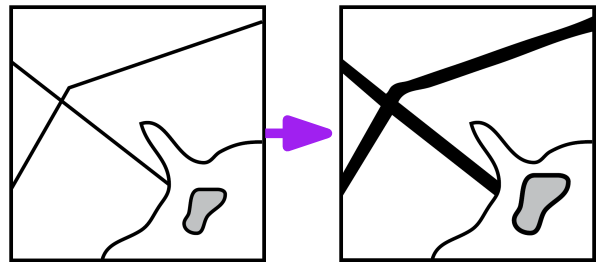
Aggregation



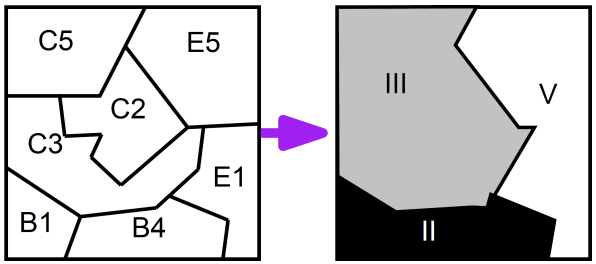
Collapse



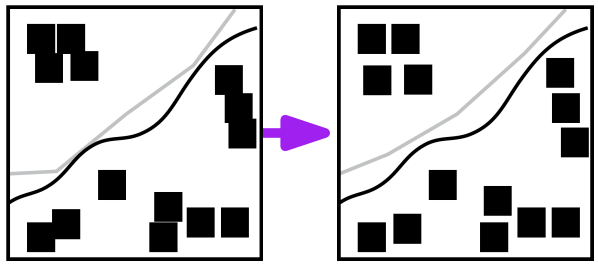
Typification



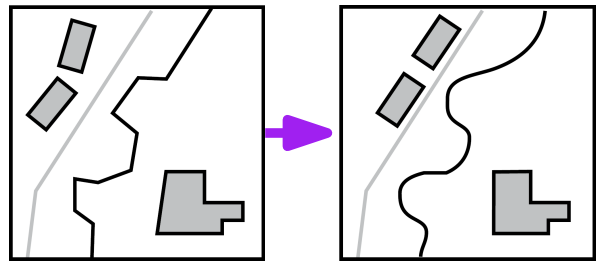
Exaggeration



Classifi. and symboli.



Displacement

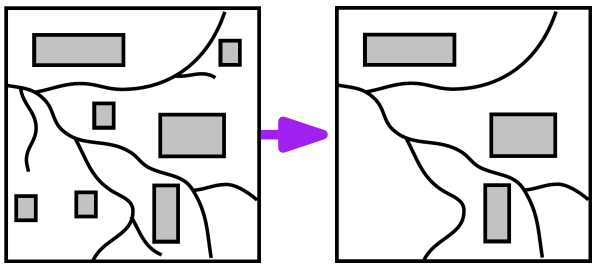


Refinement

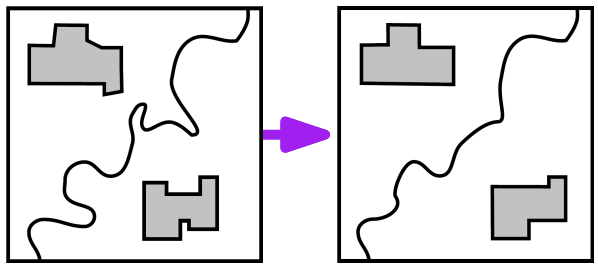
Map Generalization. . .

. . . is about selecting and representing information on a map.

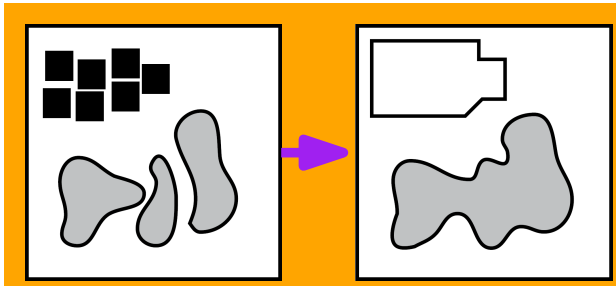
Typical generalization operators (ESRI 1996):



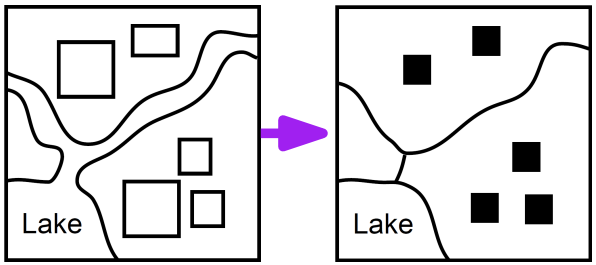
Elimination



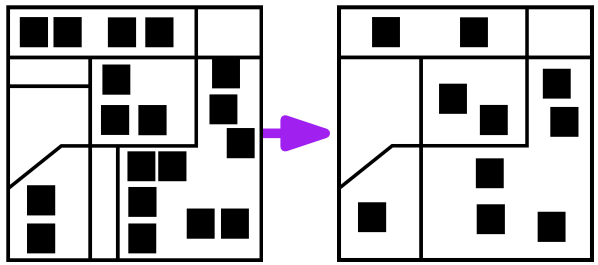
Simplification



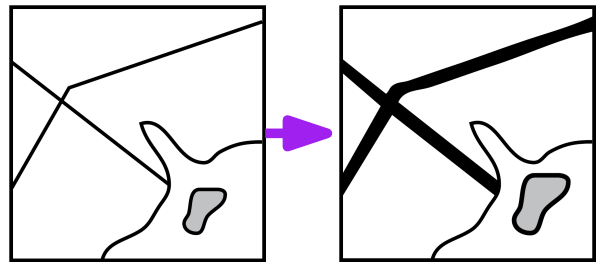
Aggregation



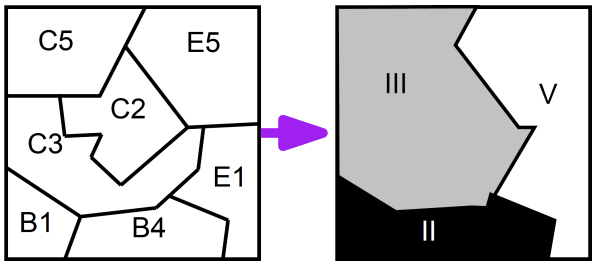
Collapse



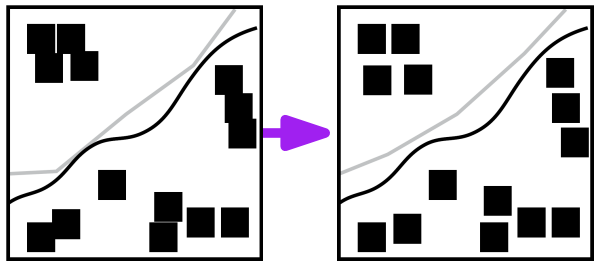
Typification



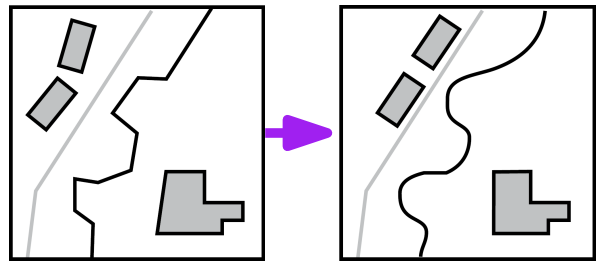
Exaggeration



Classifi. and symboli.

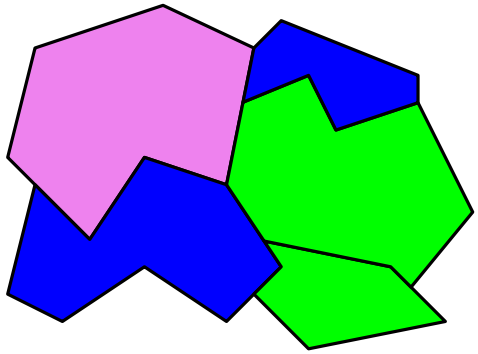


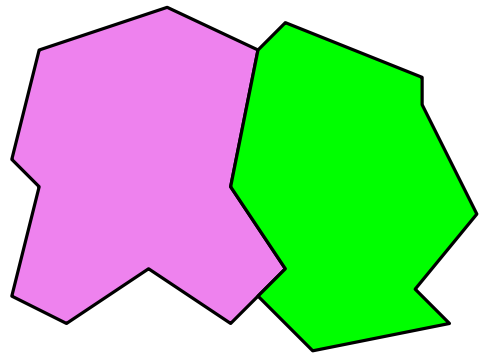
Displacement



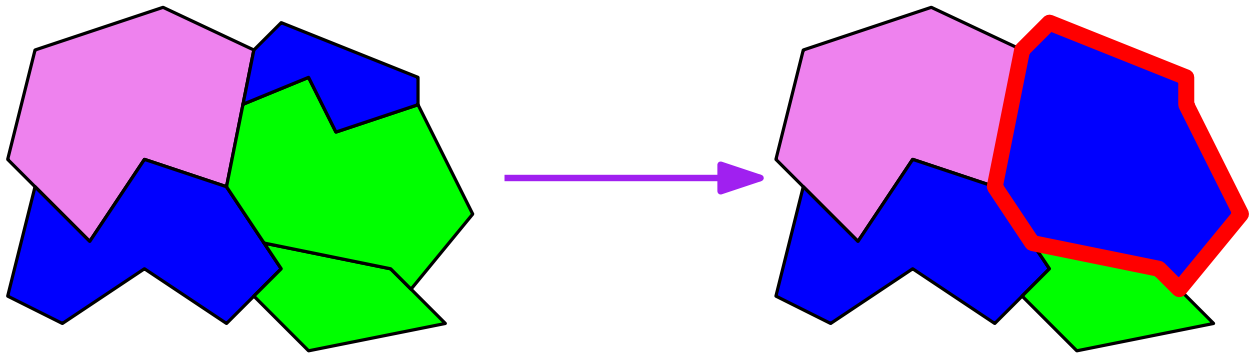
Refinement

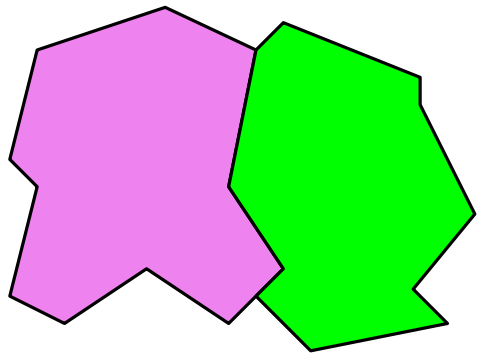
Research Problem



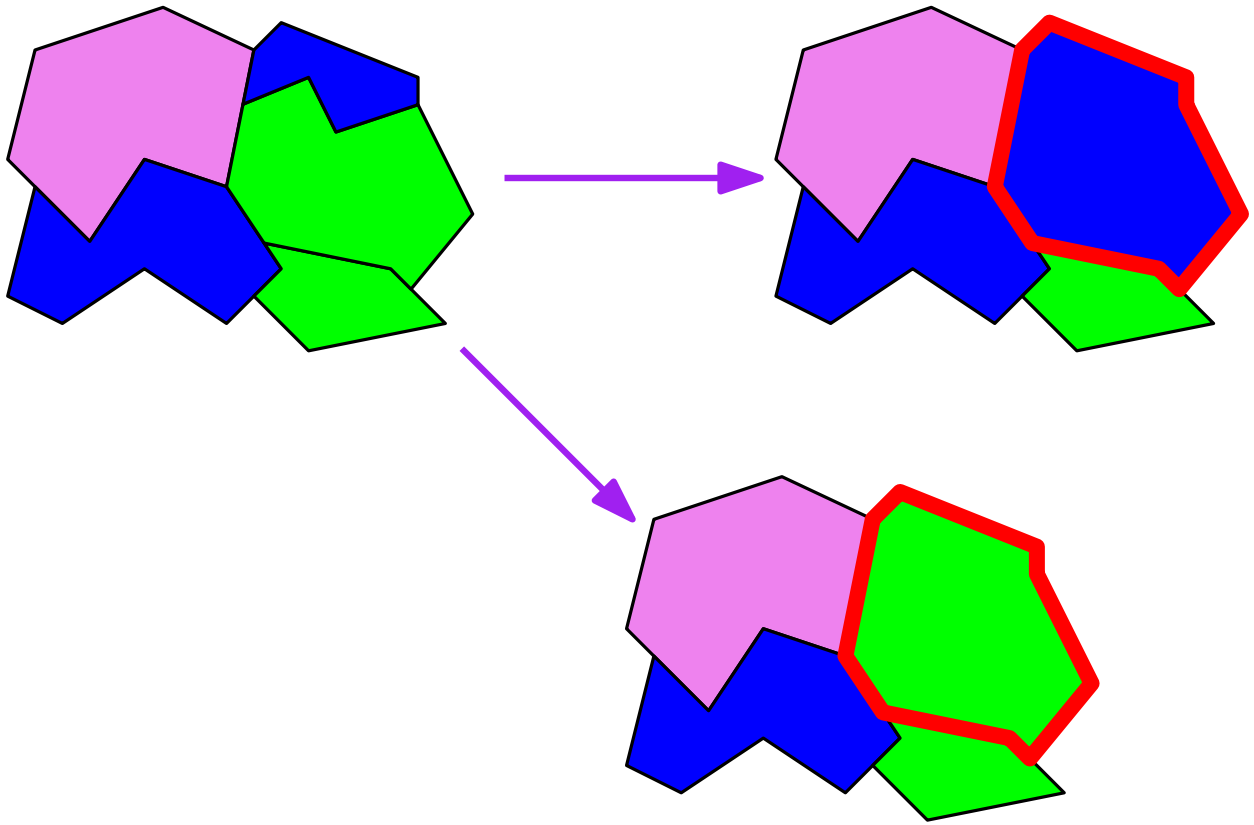


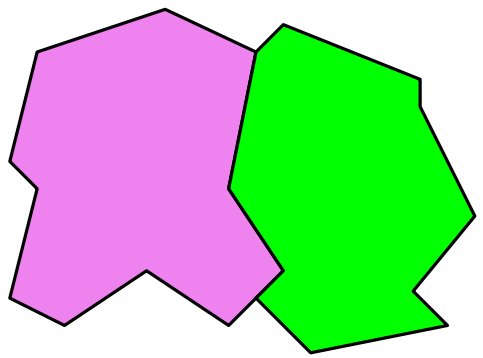
Research Problem



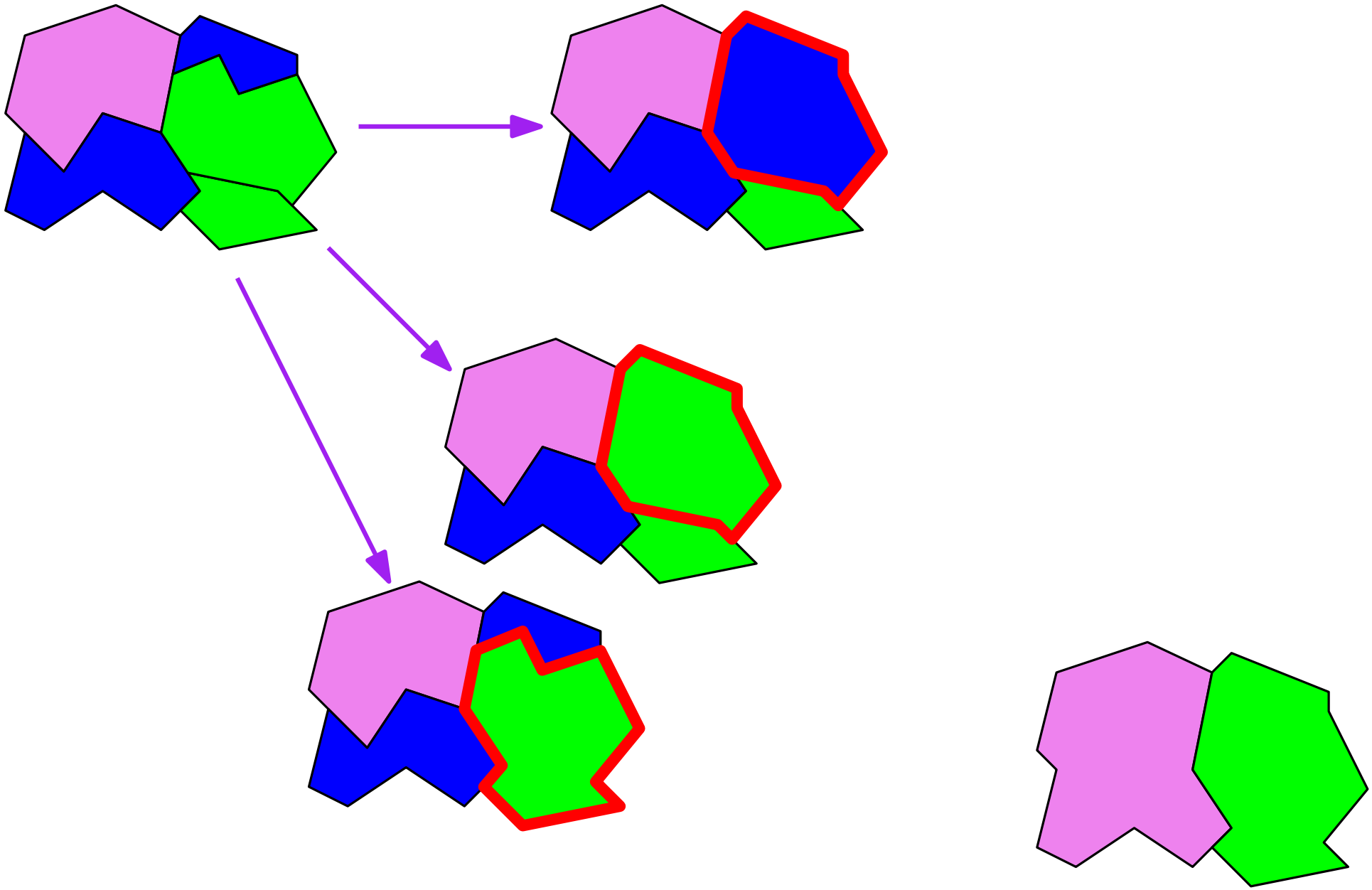


Research Problem

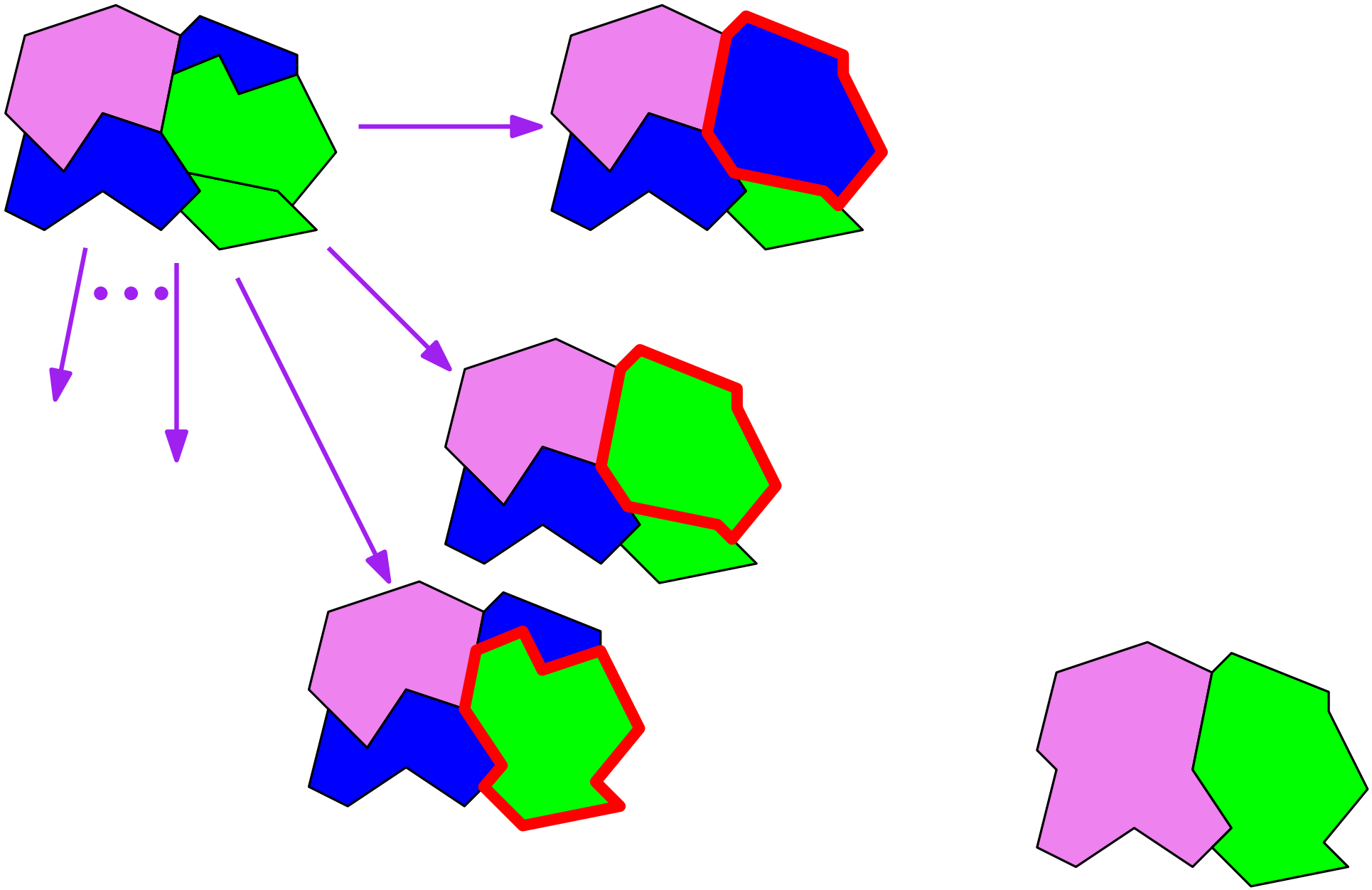




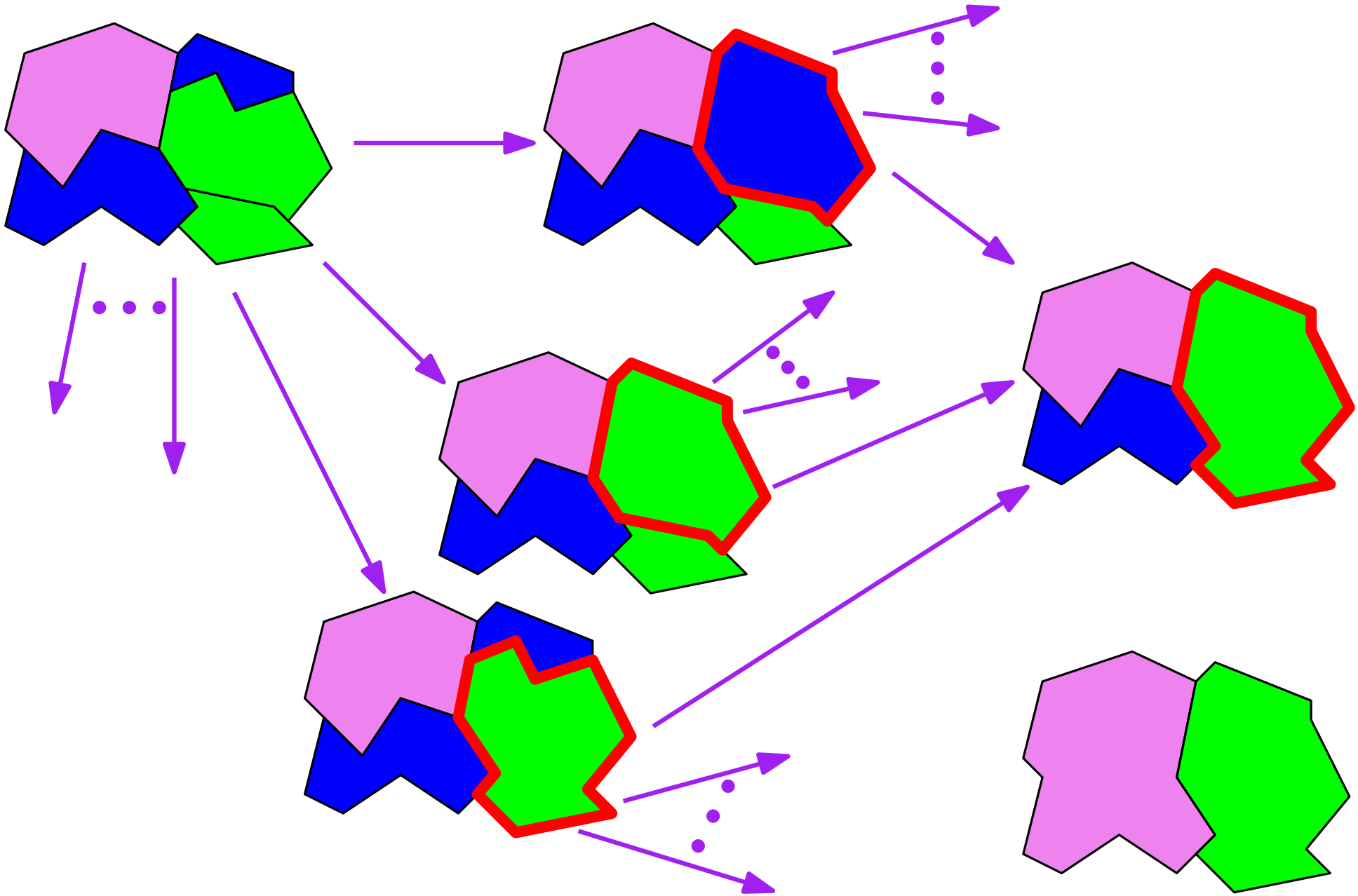
Research Problem



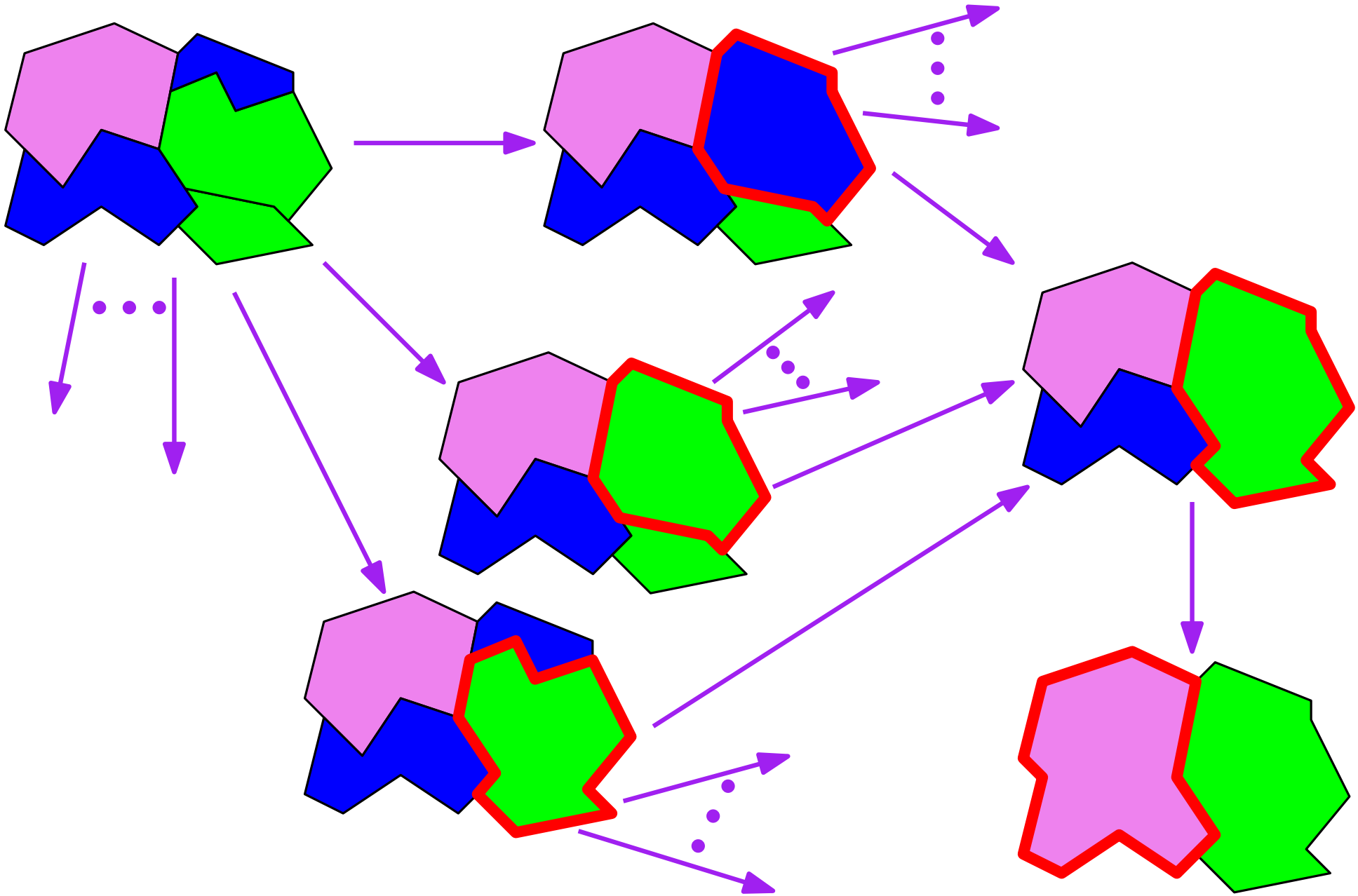
Research Problem



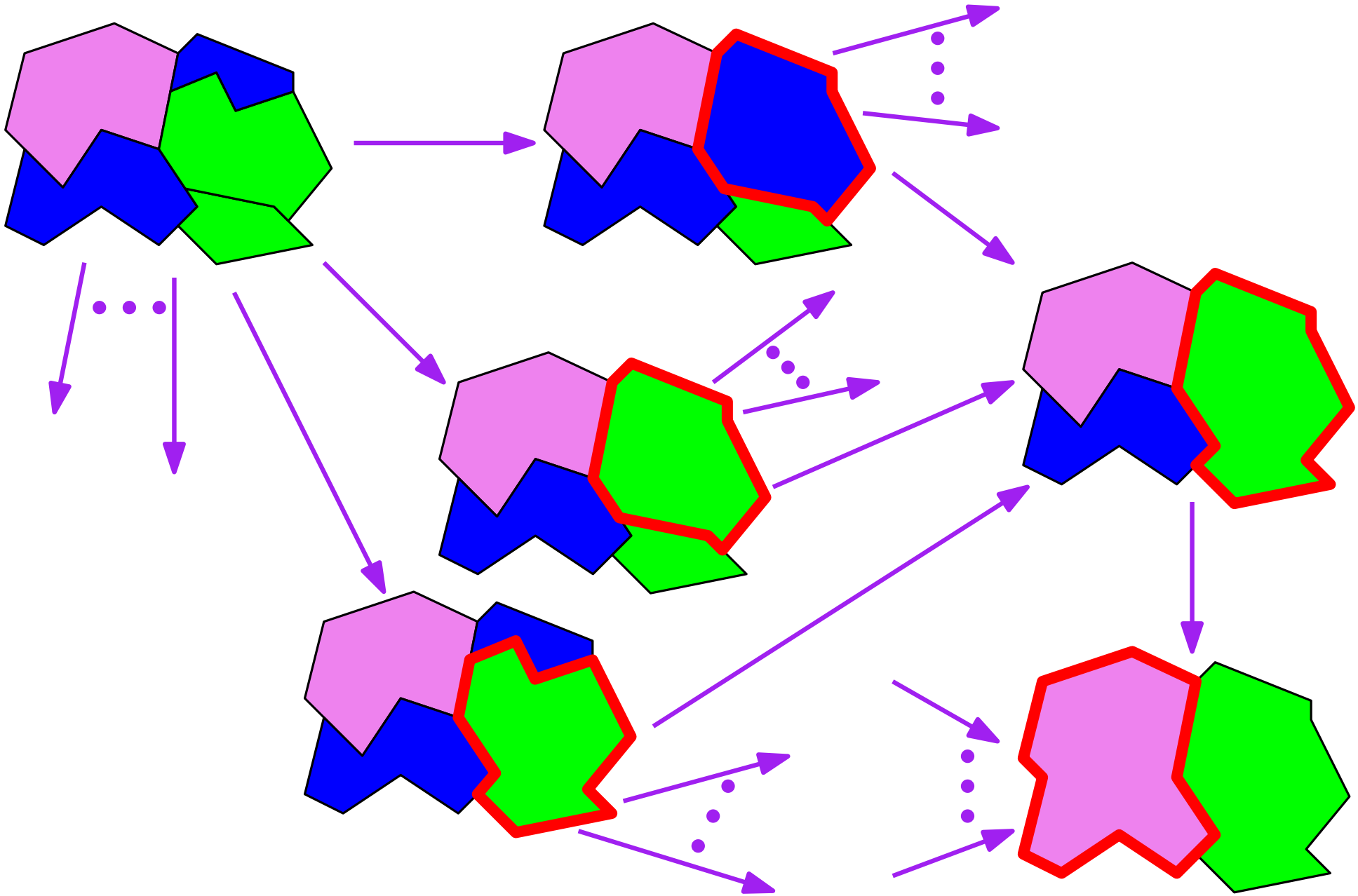
Research Problem



Research Problem



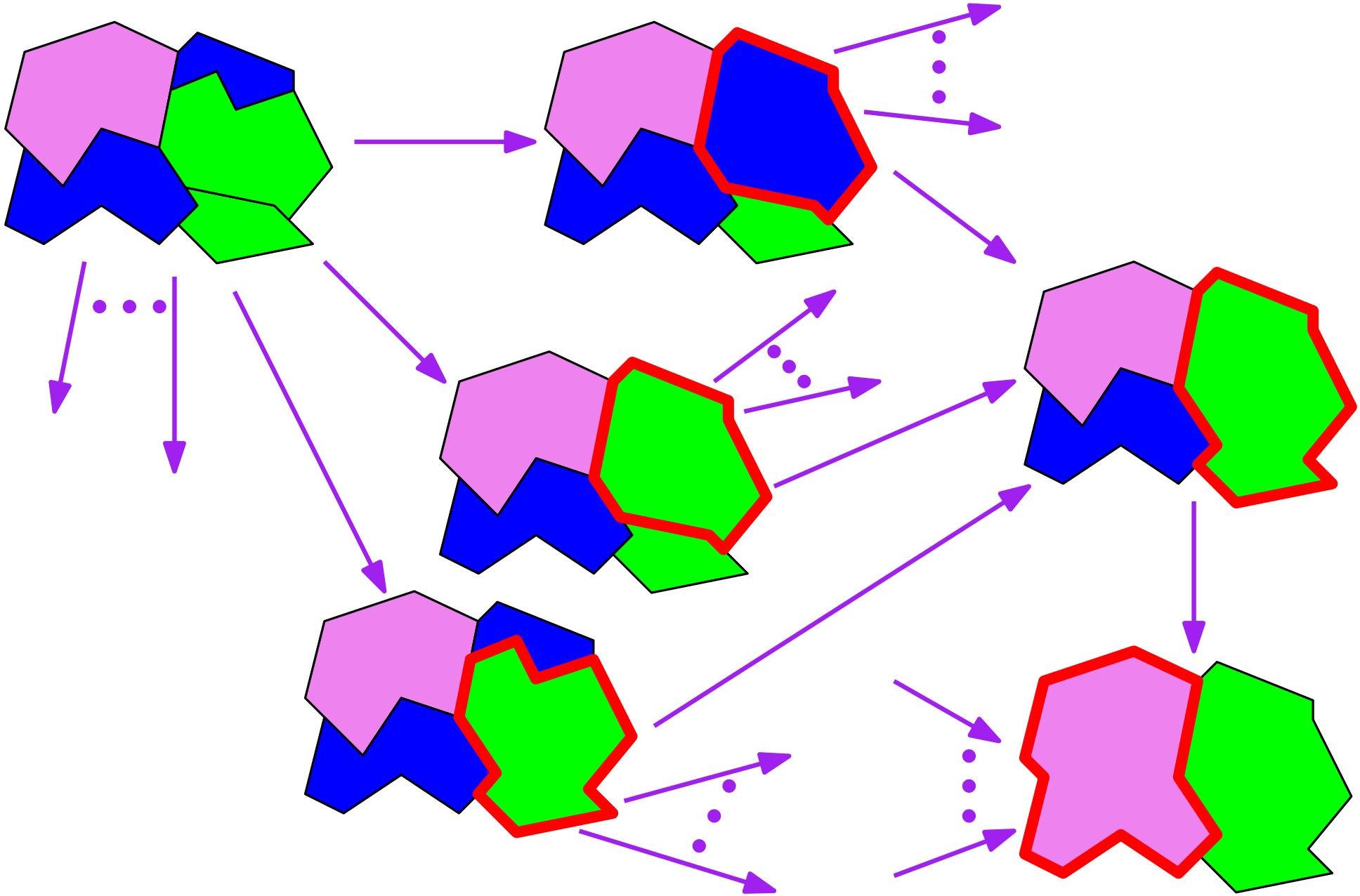
Research Problem



Research Problem

Given aggregation costs:

What is an optimal sequence?



Standard Approach for Dynamic Maps

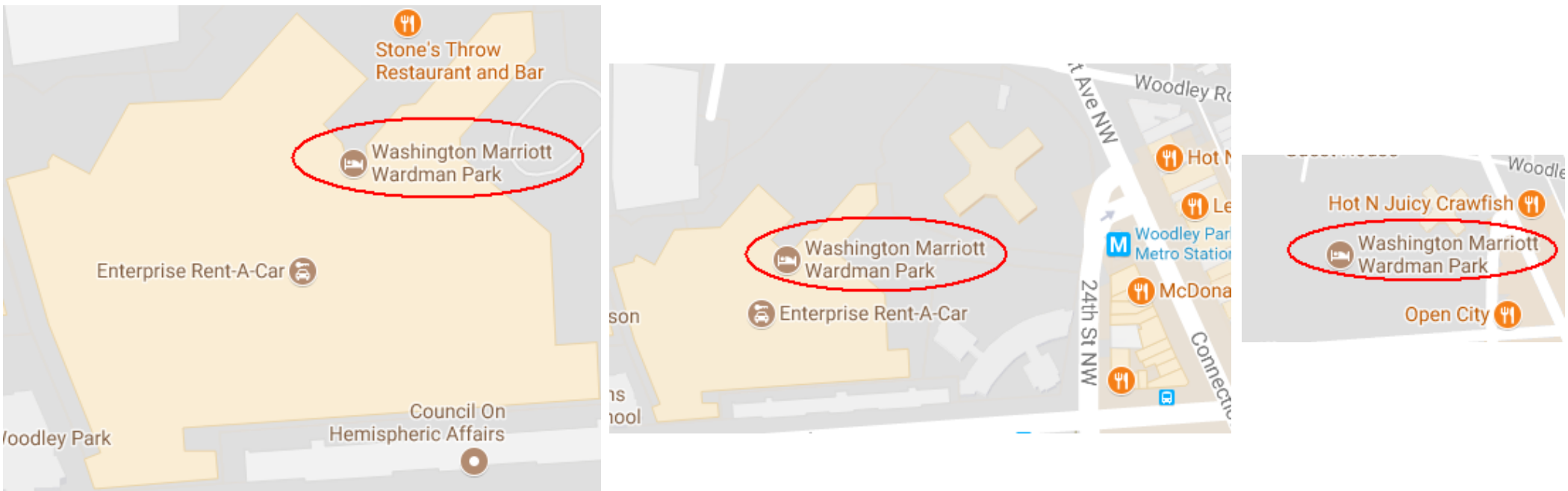
- Users zoom in and out to read digital maps

Standard Approach for Dynamic Maps

- Users zoom in and out to read digital maps
- Multi-Representation Databases (MRDB) support Levels of Detail (LODs) (Hampe et al. 2004)

Standard Approach for Dynamic Maps

- Users zoom in and out to read digital maps
- Multi-Representation Databases (MRDB) support Levels of Detail (LODs) (Hampe et al. 2004)



zoom out

(Google Maps)

Standard Approach for Dynamic Maps

- Users zoom in and out to read digital maps
- Multi-Representation Databases (MRDB) support Levels of Detail (LODs) (Hampe et al. 2004)

Expect continuous change!

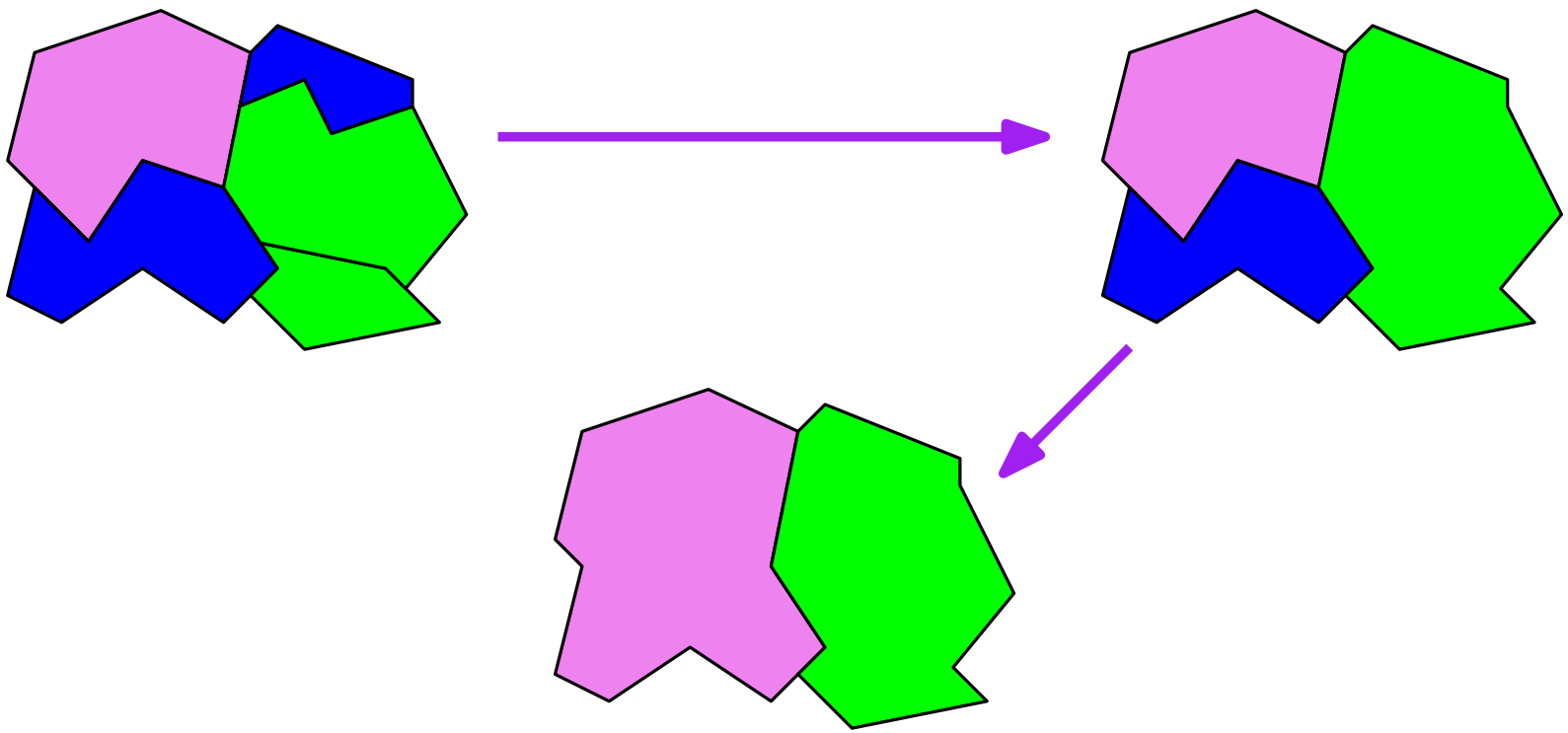
zoom out

(Google Maps)

Outline

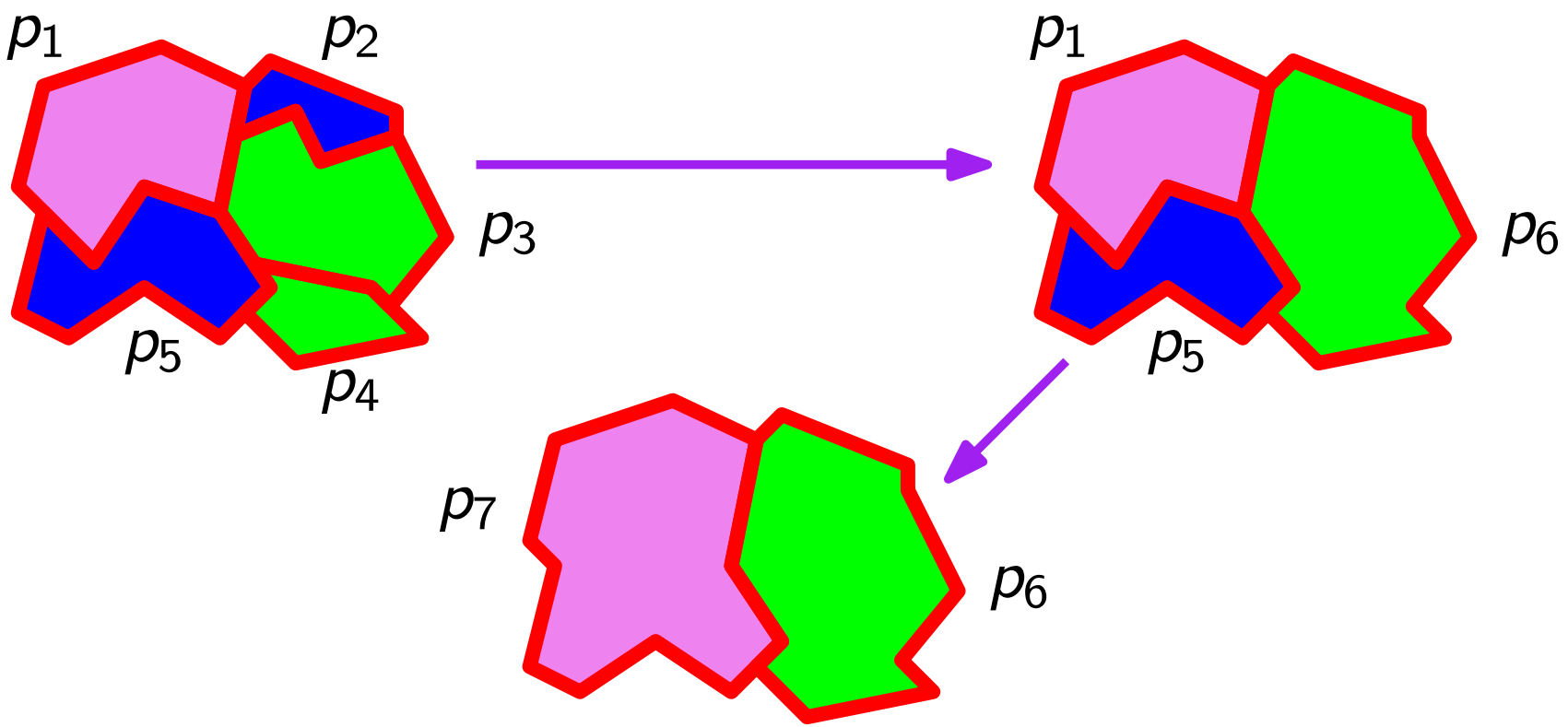
- Introduction
- Methodology
- Case Study
- Concluding Remarks

Preliminaries



Preliminaries

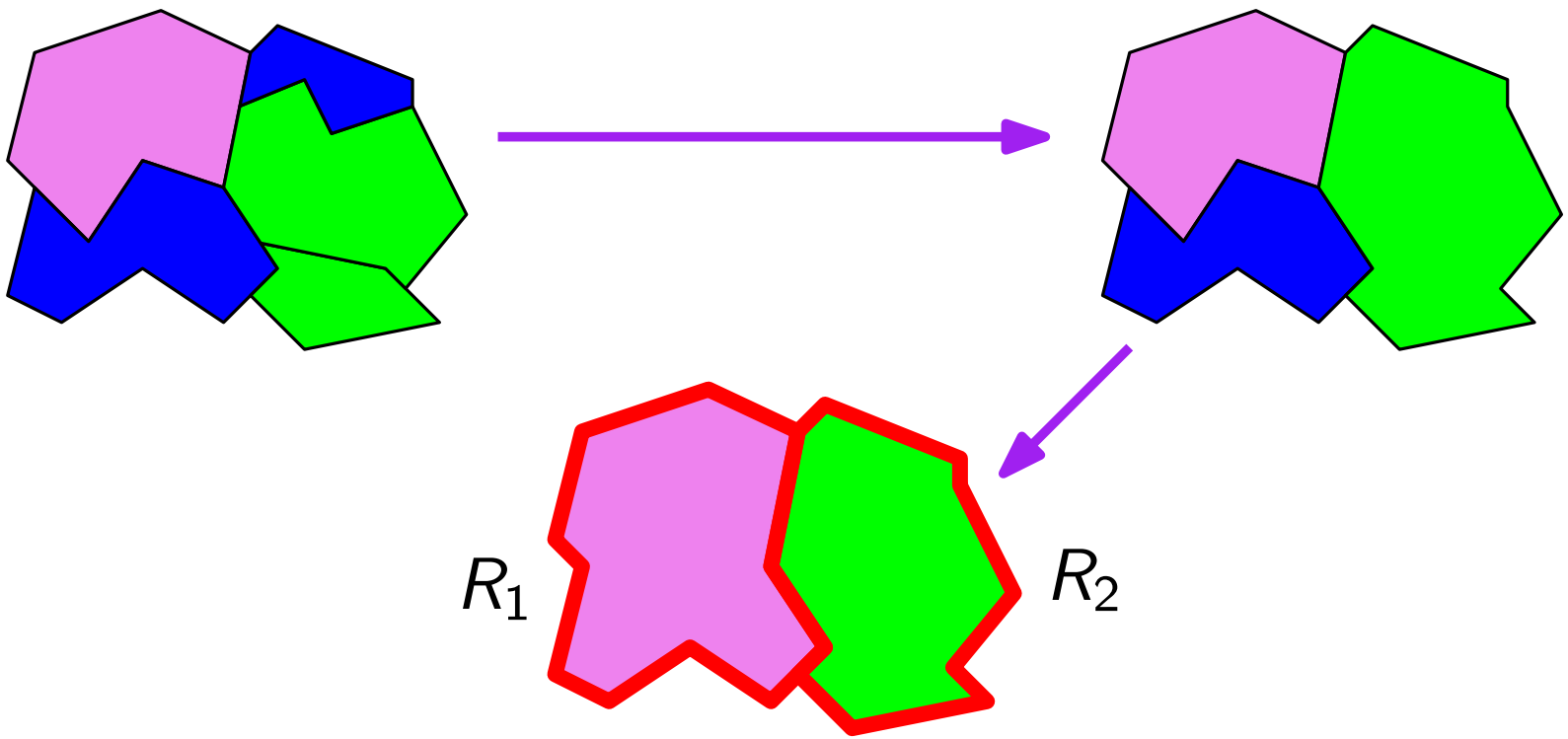
Patch p_i: a set of land-cover areas whose union is connected



Preliminaries

Patch p_i: a set of land-cover areas whose union is connected

Region R_i: an area on the goal map

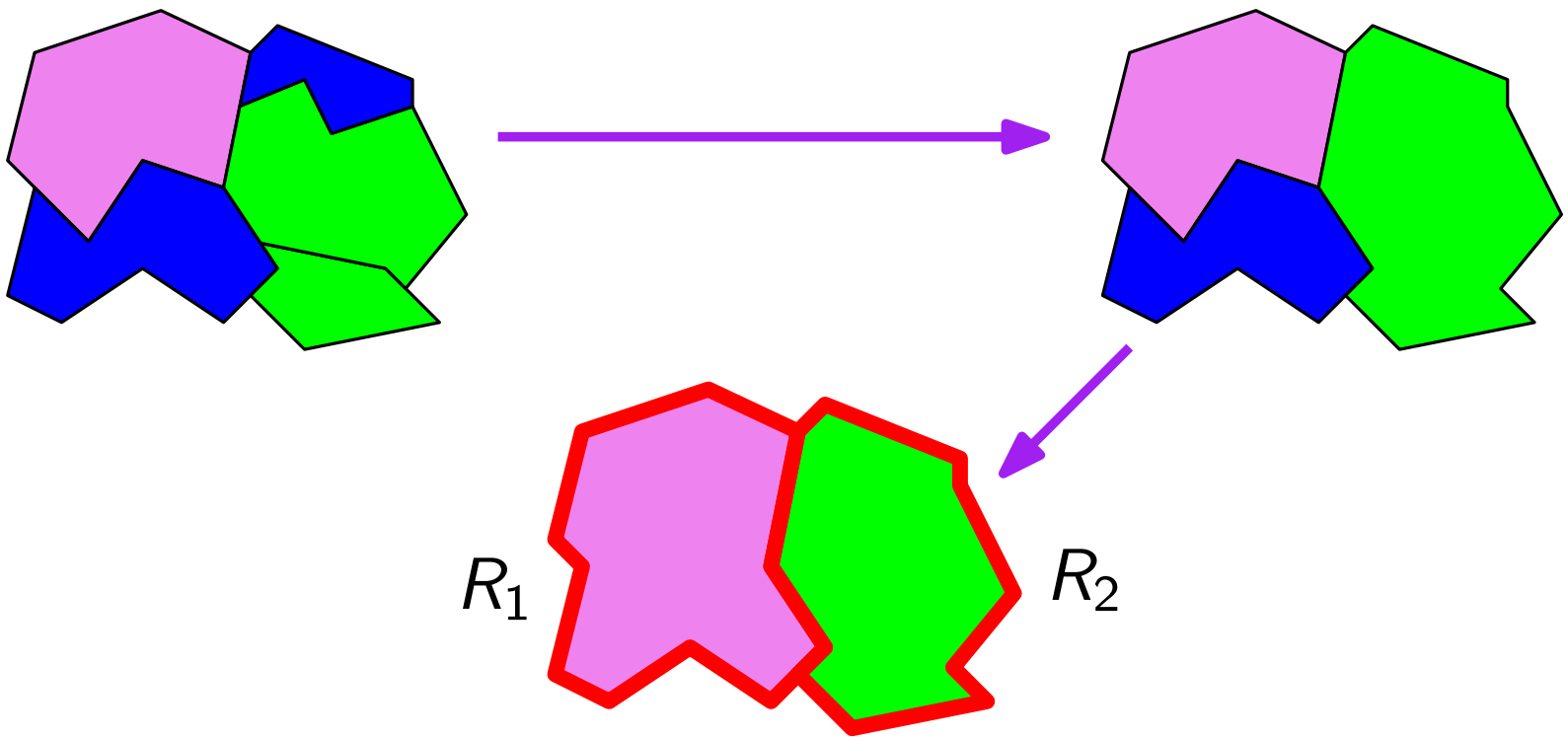


Preliminaries

Patch p_i: a set of land-cover areas whose union is connected

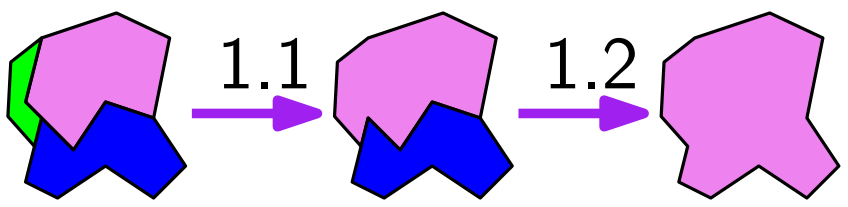
Region R_i: an area on the goal map

Aggregate the smallest patch with one of its neighbours at each step

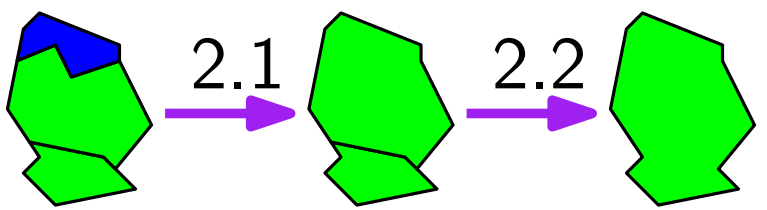


Integrate Aggregation Sequences

according to the order of smallest areas



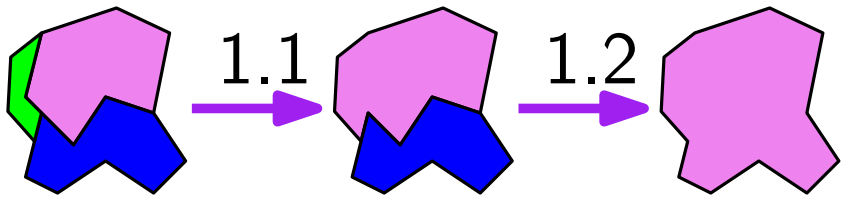
sequence of the pink region



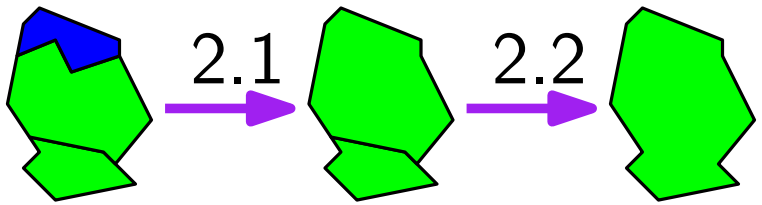
sequence of the green region

Integrate Aggregation Sequences

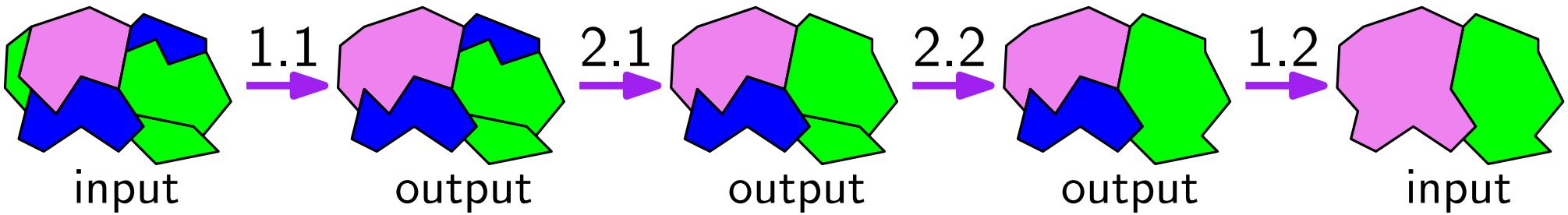
according to the order of smallest areas



sequence of the pink region



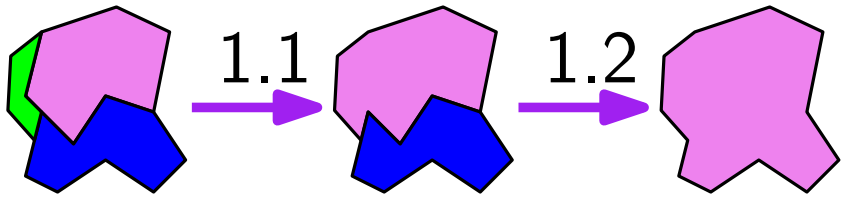
sequence of the green region



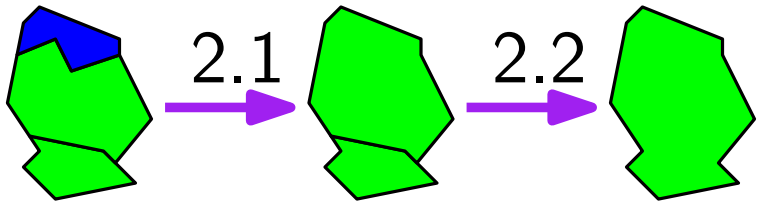
integrated sequence of the two regions

Integrate Aggregation Sequences

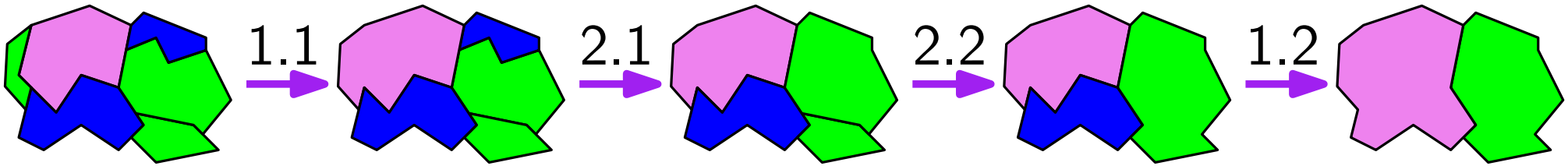
according to the order of smallest areas



sequence of the pink region



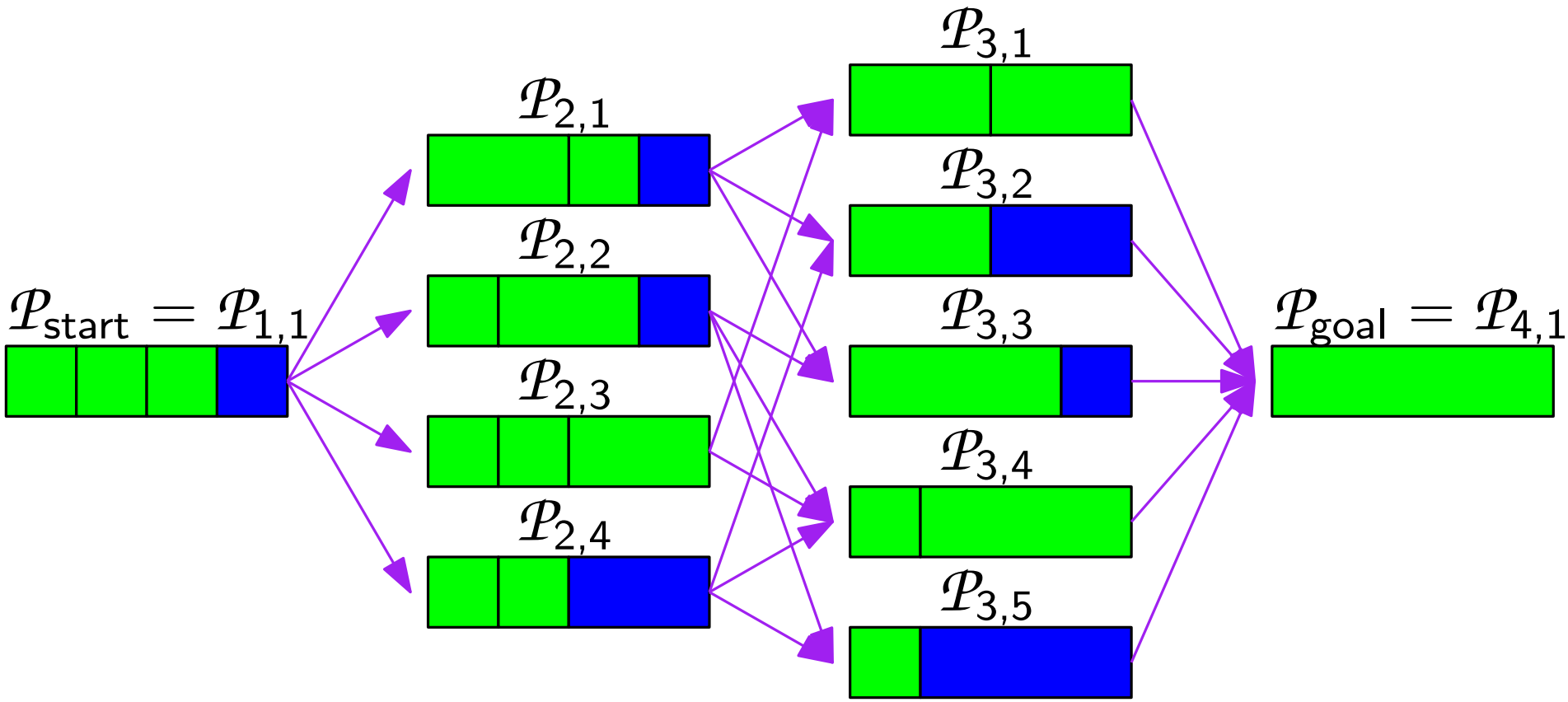
sequence of the green region



integrated sequence of the two regions

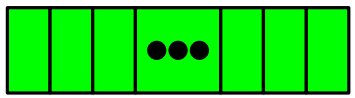
Subdivision

Subdivision $\mathcal{P}_{t,i}$: a set of patches subdividing a single region R



Number of Subdivisions

layer 1

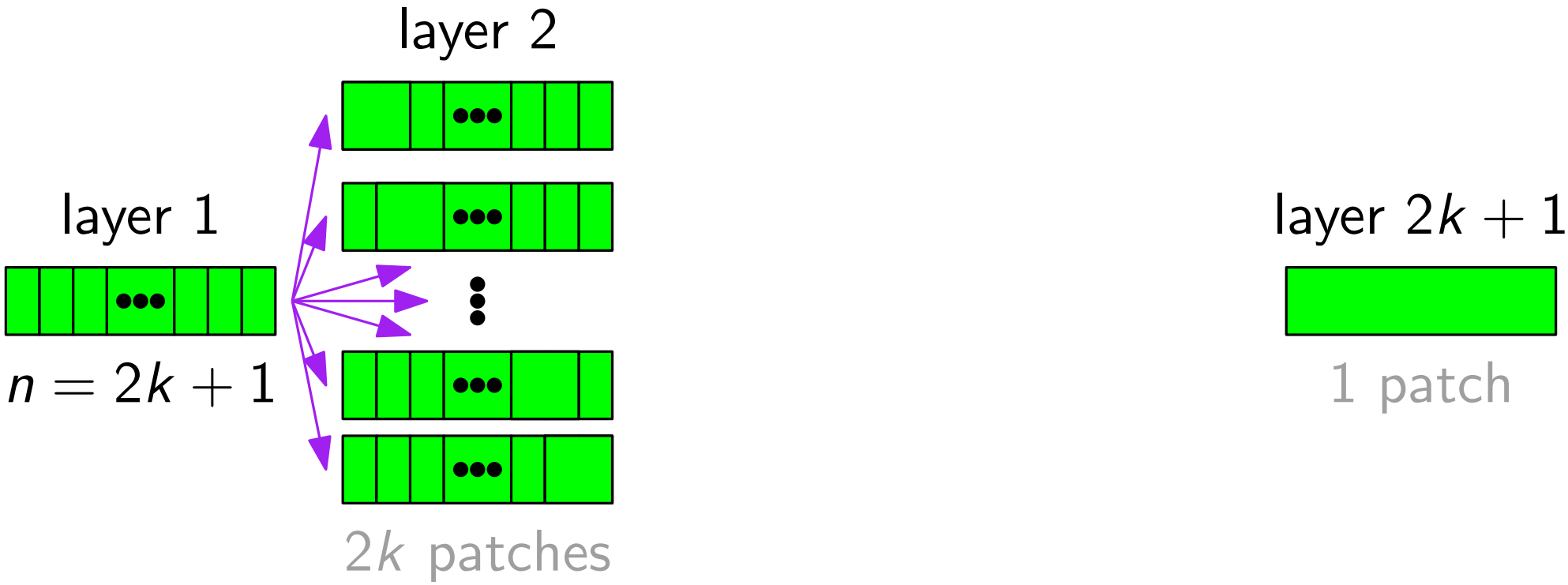


$n = 2k + 1$

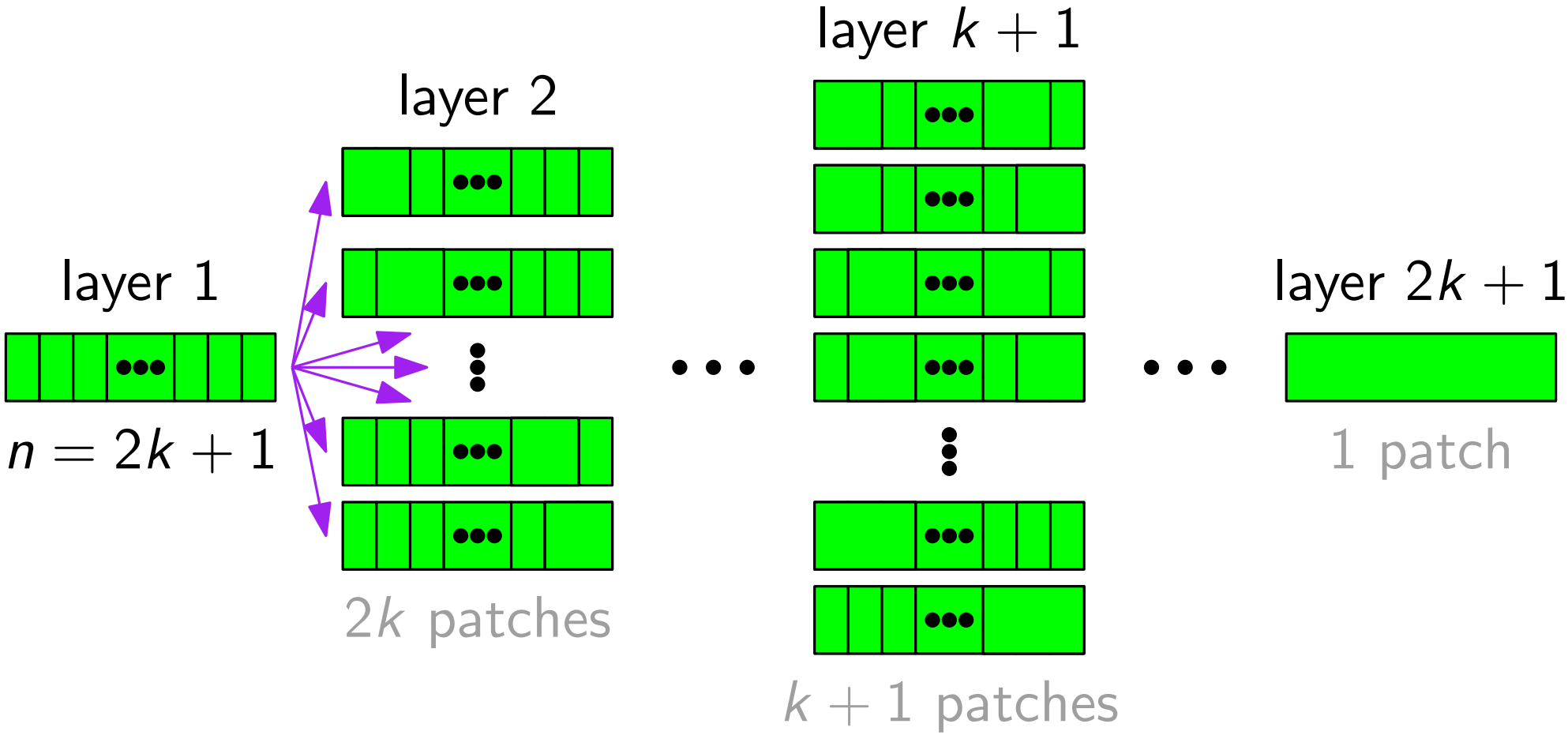
layer $2k + 1$

1 patch

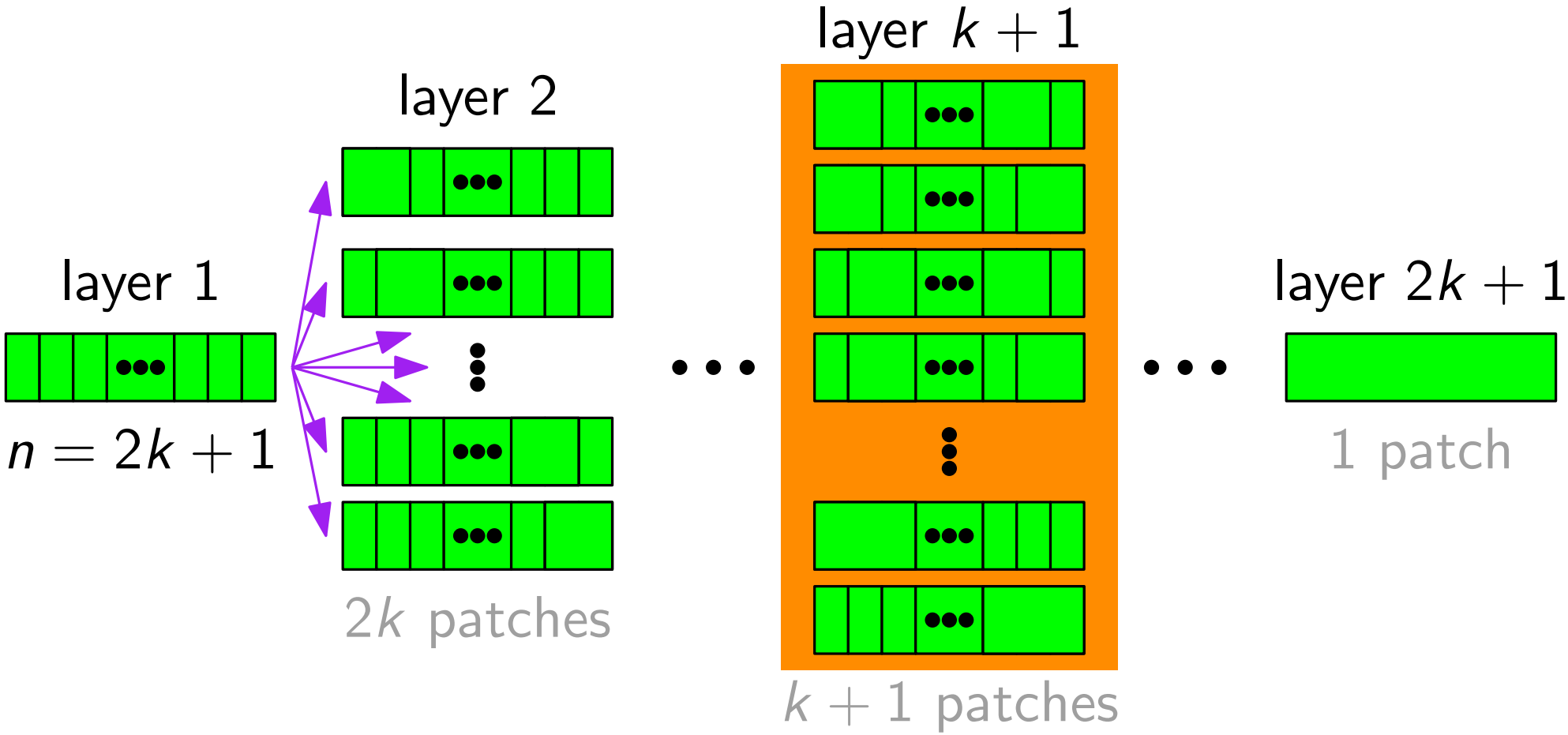
Number of Subdivisions



Number of Subdivisions

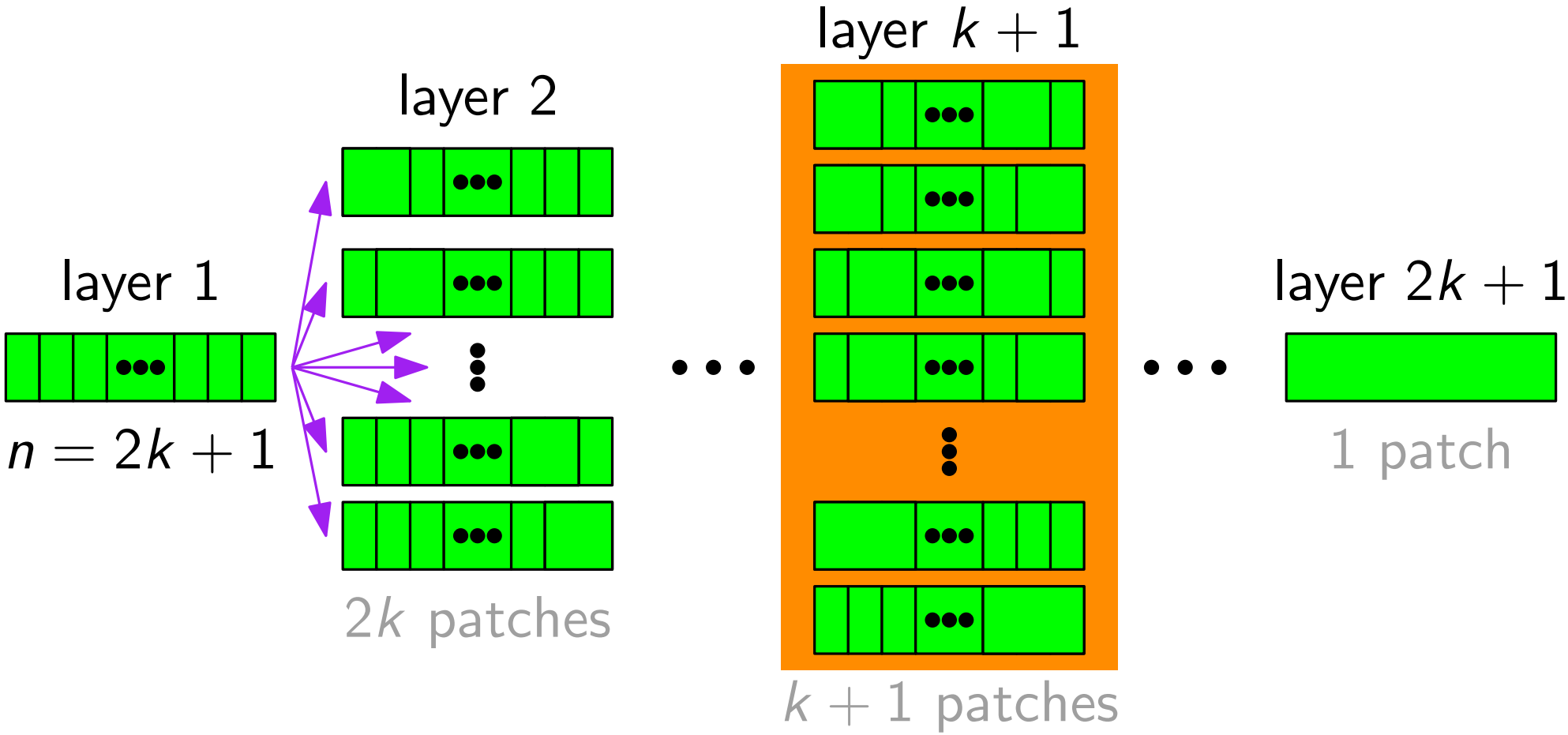


Number of Subdivisions



At layer $k+1$, remove k of the $2k$ intermediate sticks:
$\binom{2k}{k} \geq 2^k = 2^{(n-1)/2}$ subdivisions

Number of Subdivisions

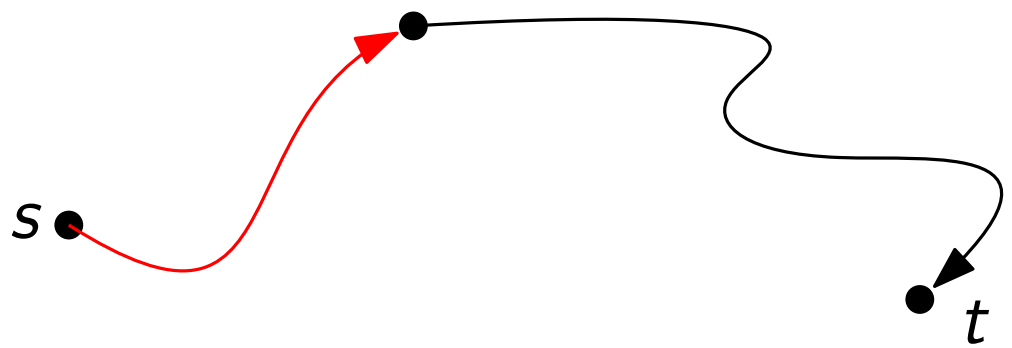


At layer $k+1$, remove k of the $2k$ intermediate sticks:
$\binom{2k}{k} \geq 2^k = 2^{(n-1)/2}$ subdivisions

$\Rightarrow$ The number of subdivions is at least *exponential* in the number n of polygons in layer $k+1$.

A* Algorithm

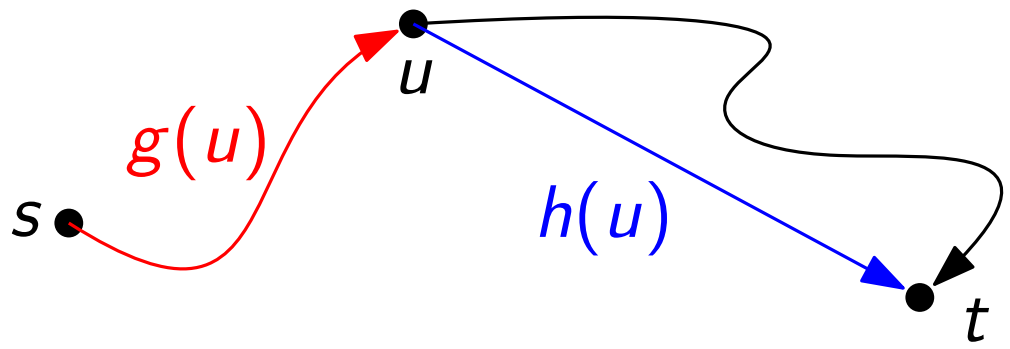
- A best-first search algorithm to find a path from a source node s to a target node t



A* Algorithm

- A best-first search algorithm to find a path from a source node s to a target node t

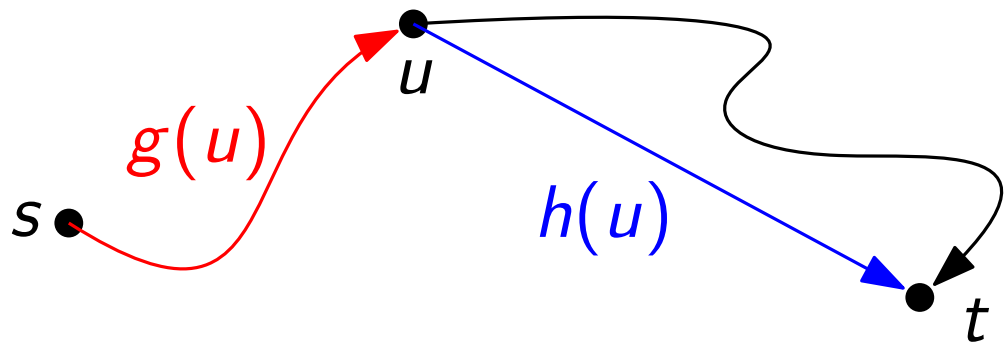
- Cost function: $F(u) = g(u) + h(u)$
  - $g(u)$: exact cost of s-u path
  - $h(u)$: estimated cost of shortest u-t path



A* Algorithm

- A best-first search algorithm to find a path from a source node s to a target node t

- Cost function: $F(u) = g(u) + h(u)$
  - $g(u)$: exact cost of s-u path
  - $h(u)$: estimated cost of shortest u-t path



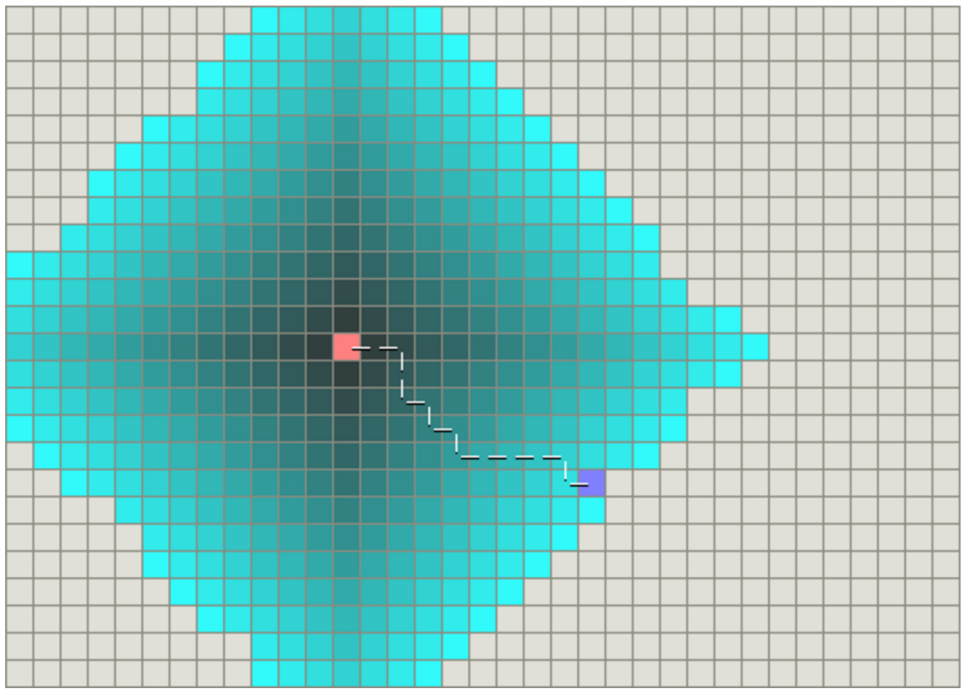
- Guarantees a shortest path if $h(u)$ never overestimates.

Shortest-Path Algorithms

Find shortest path from pink cell to blue cell

(Amit's A* Pages)

Dijkstra's algorithm:

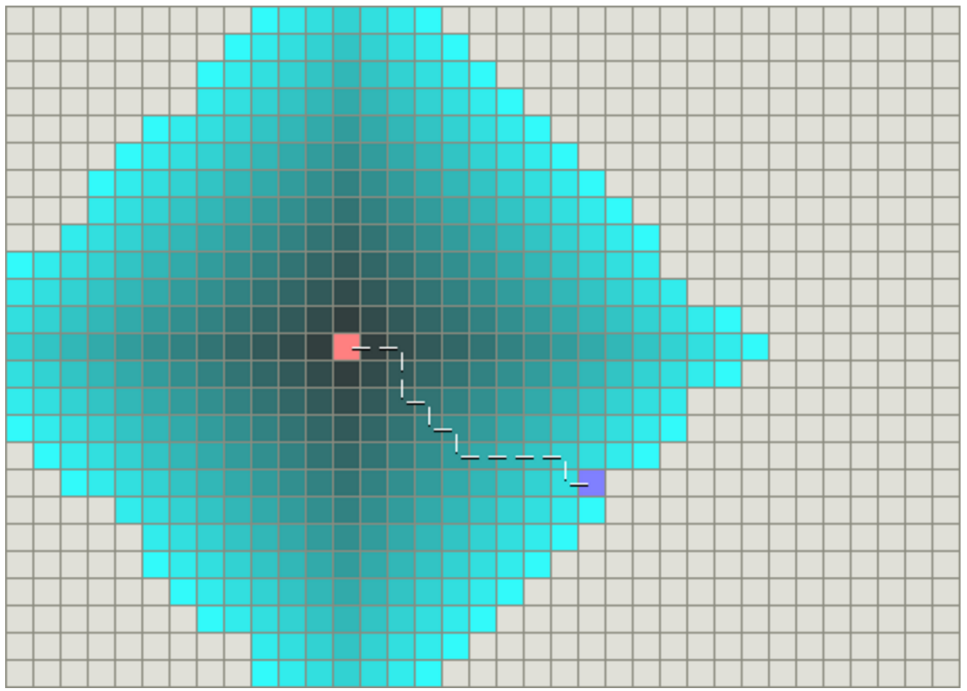


Shortest-Path Algorithms

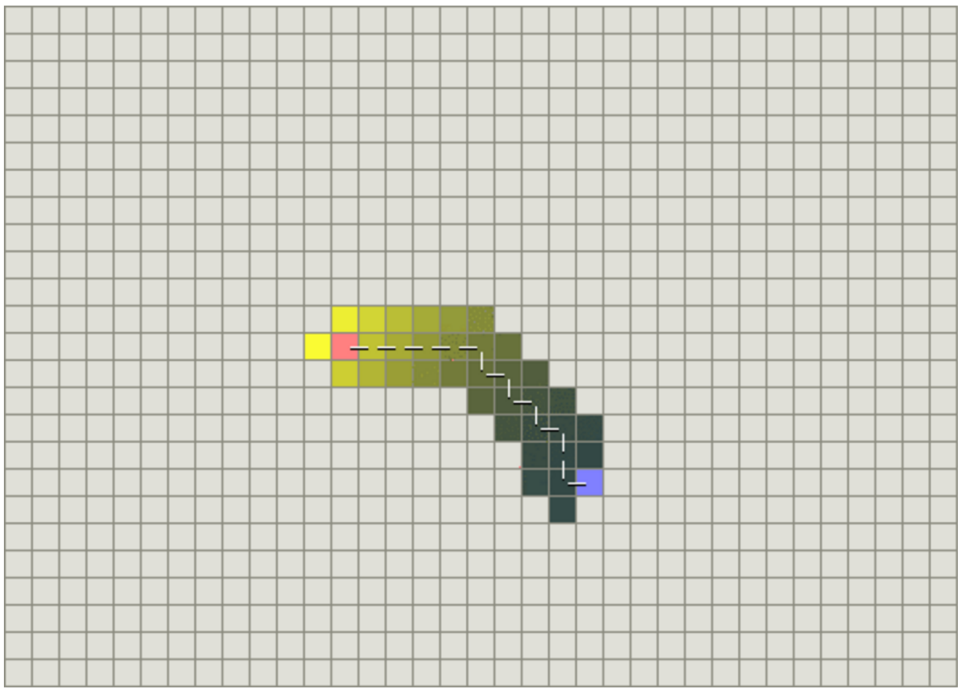
Find shortest path from pink cell to blue cell

(Amit's A* Pages)

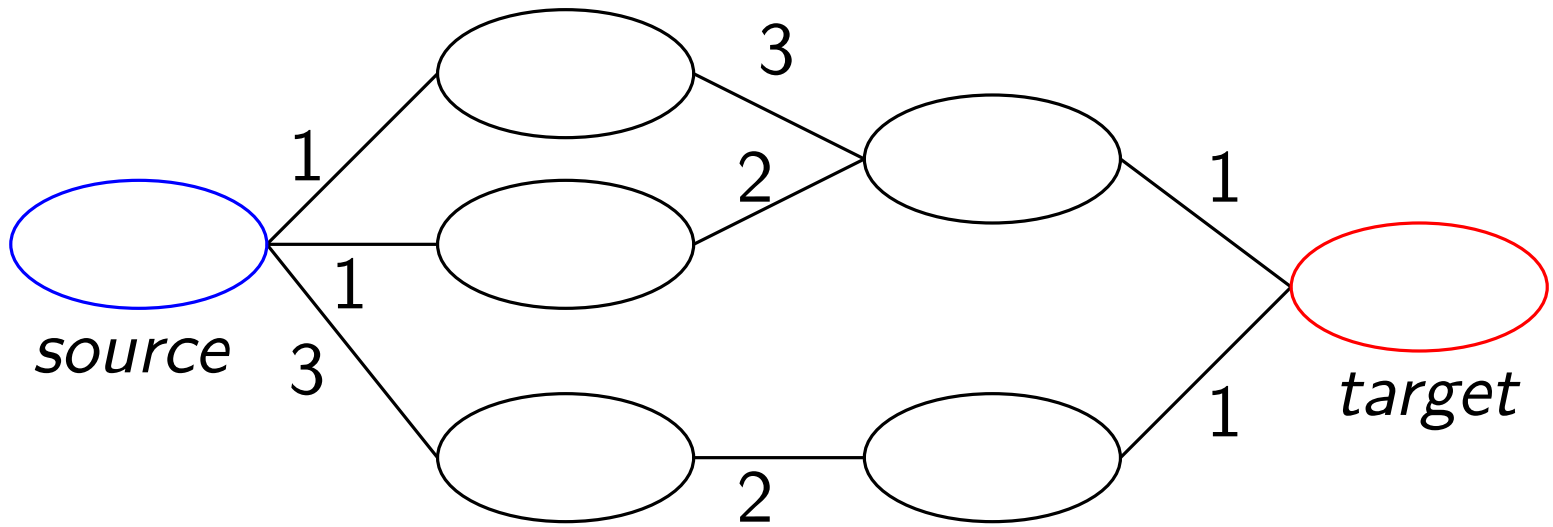
Dijkstra's algorithm:



A* algorithm:

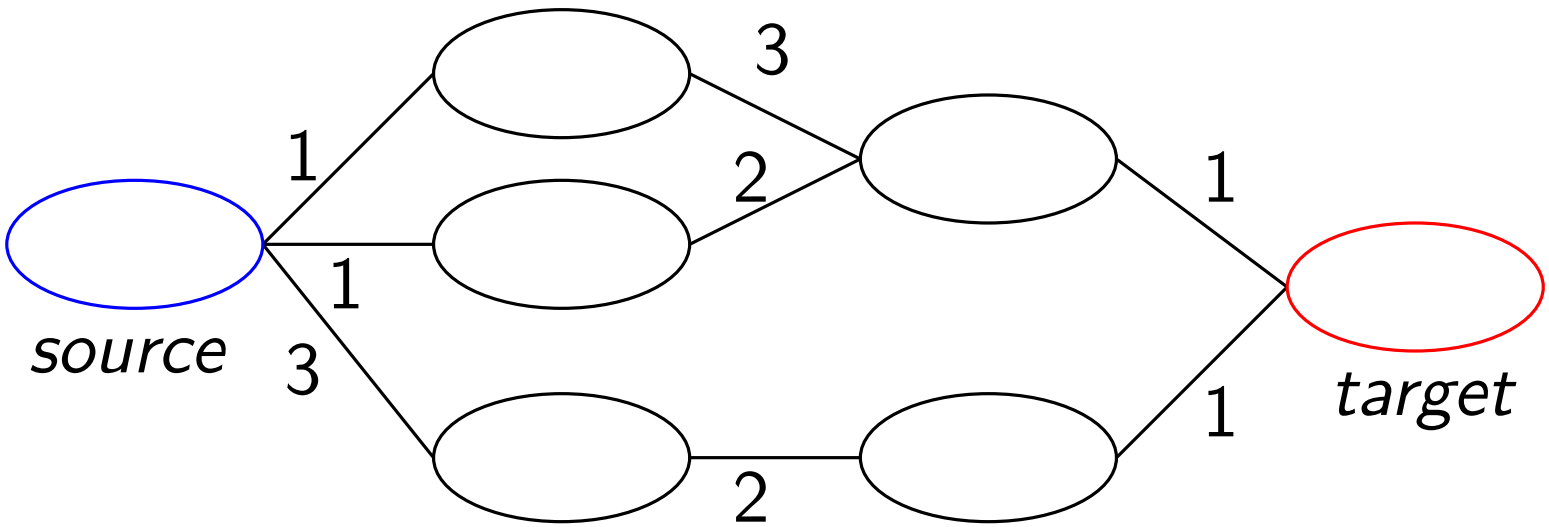


A* Algorithm

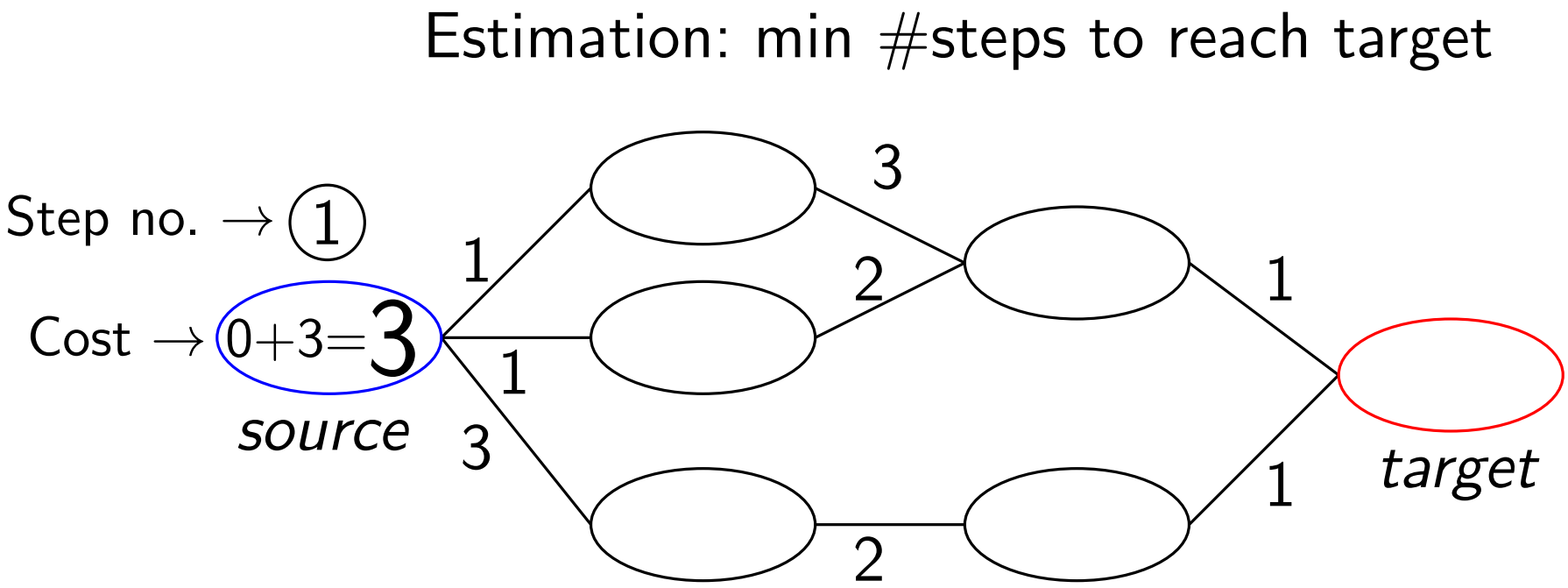


A* Algorithm

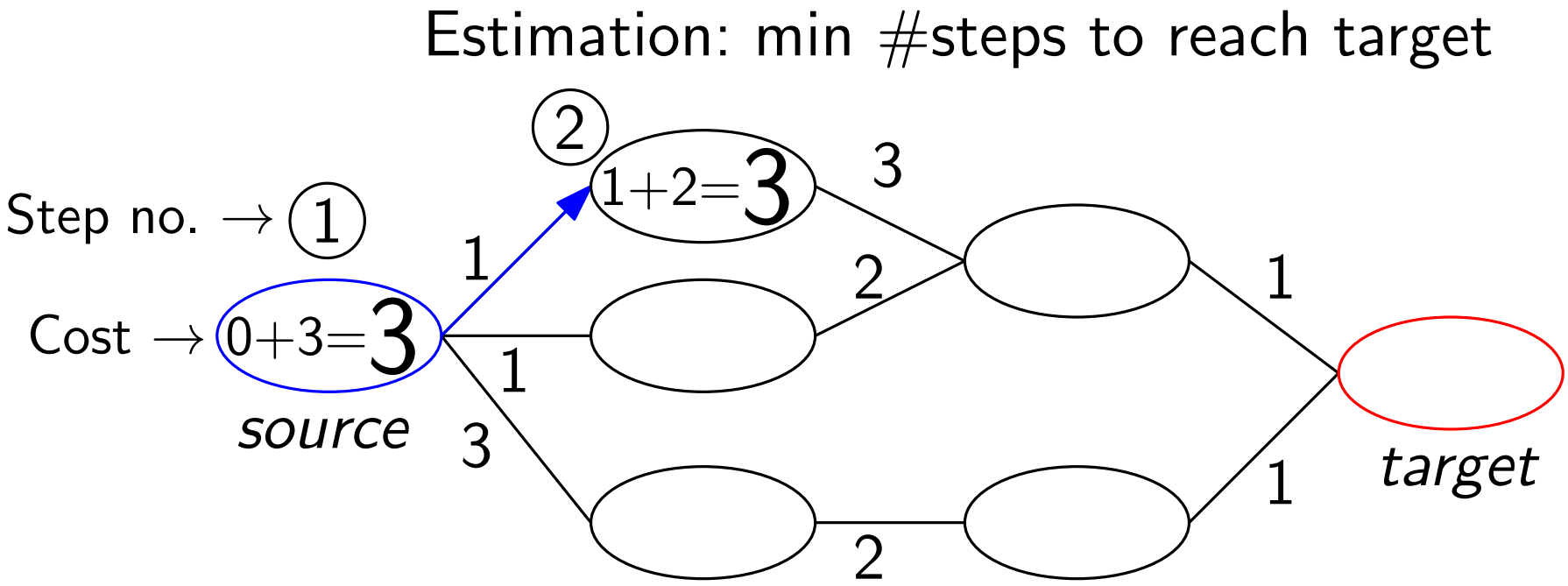
Estimation: min #steps to reach target



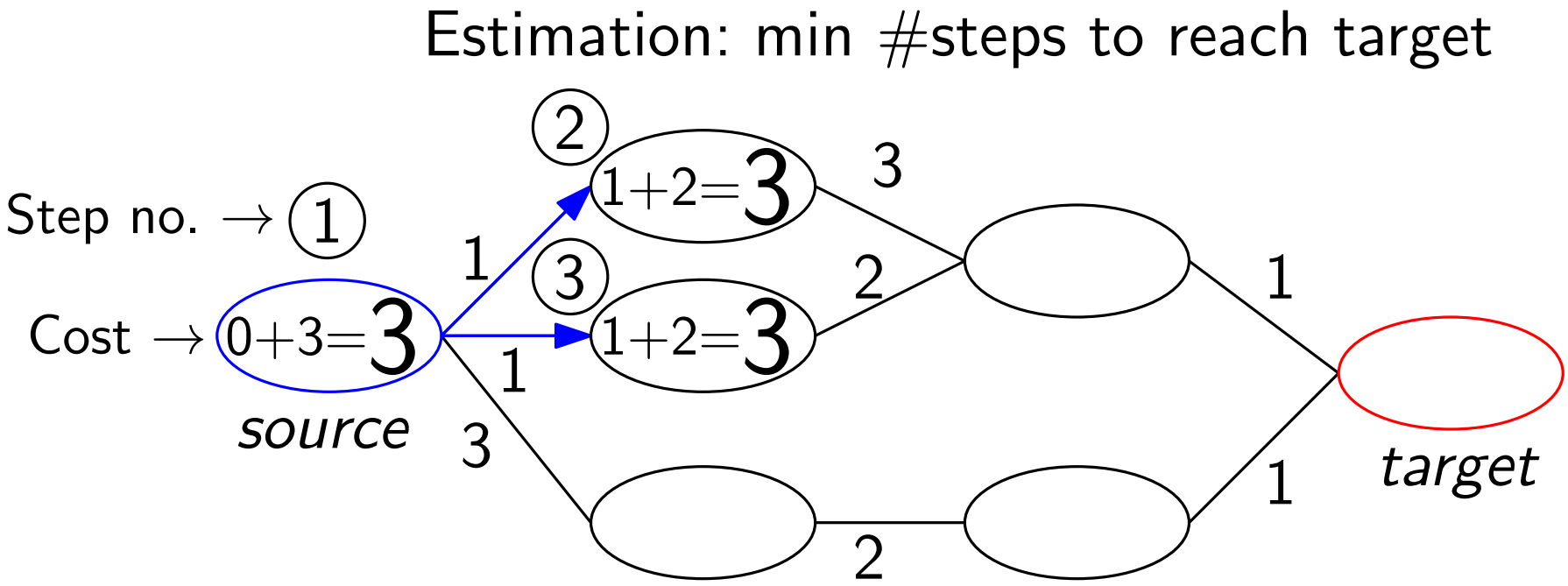
A* Algorithm



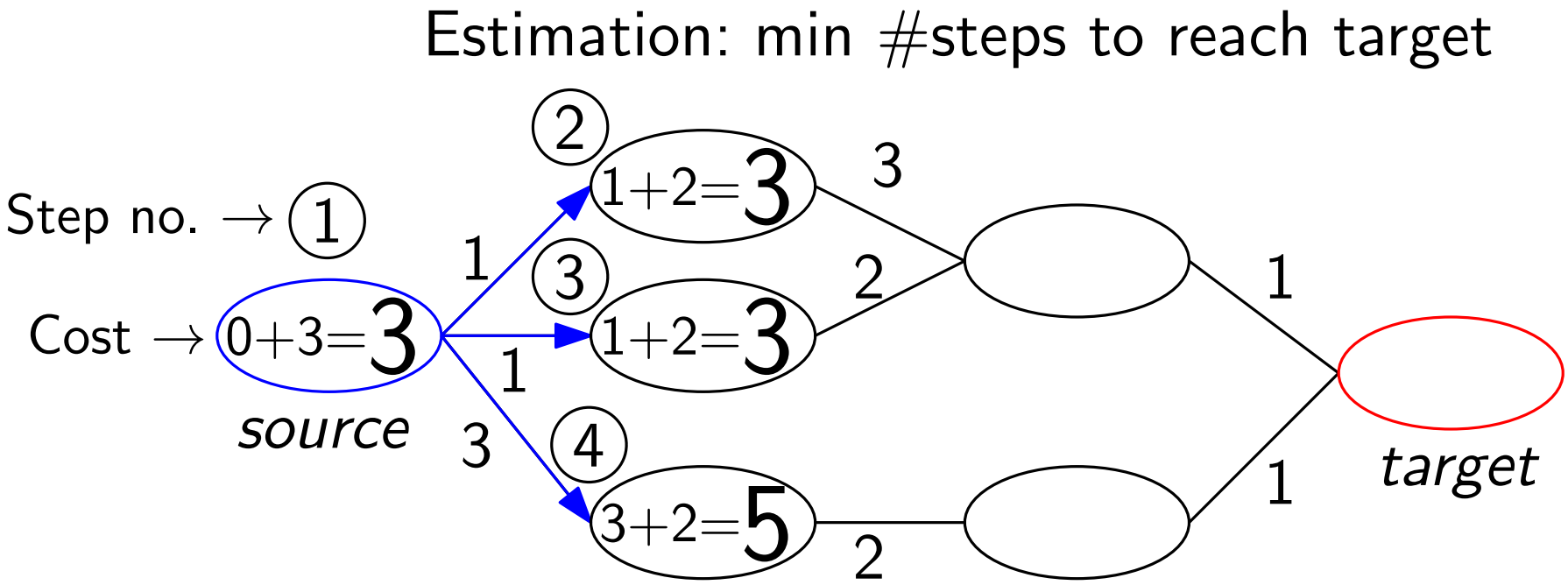
A* Algorithm



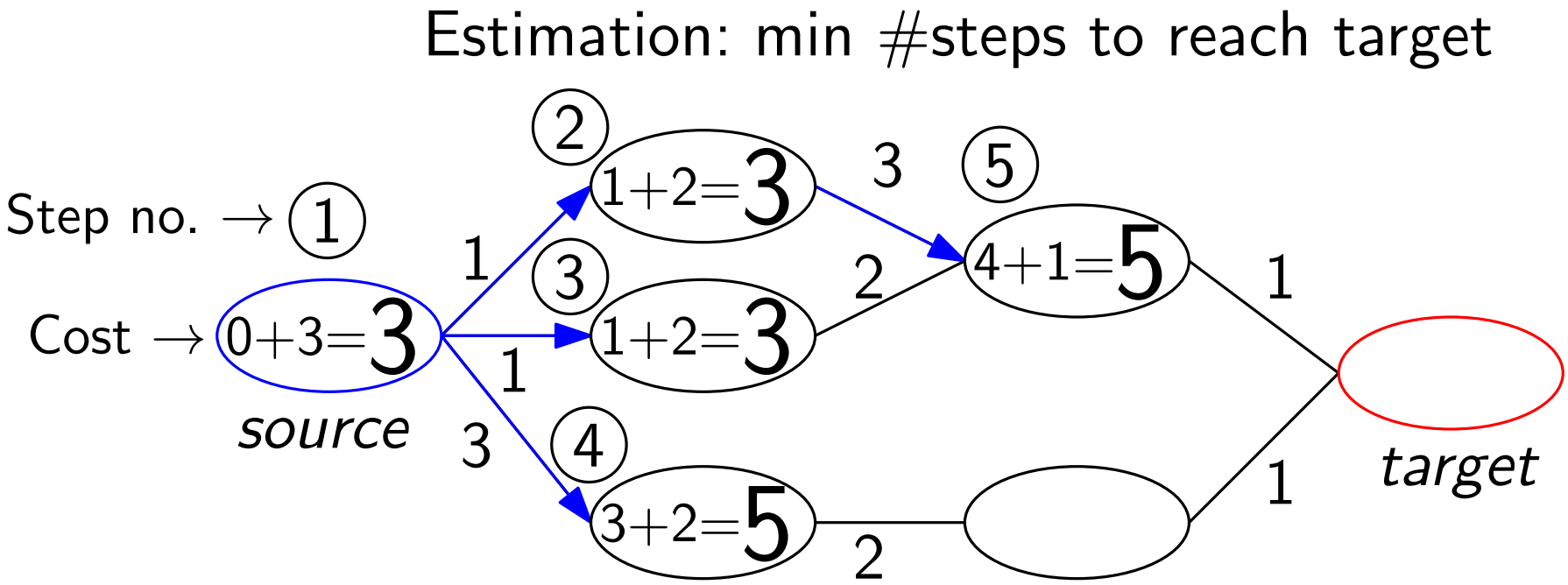
A* Algorithm



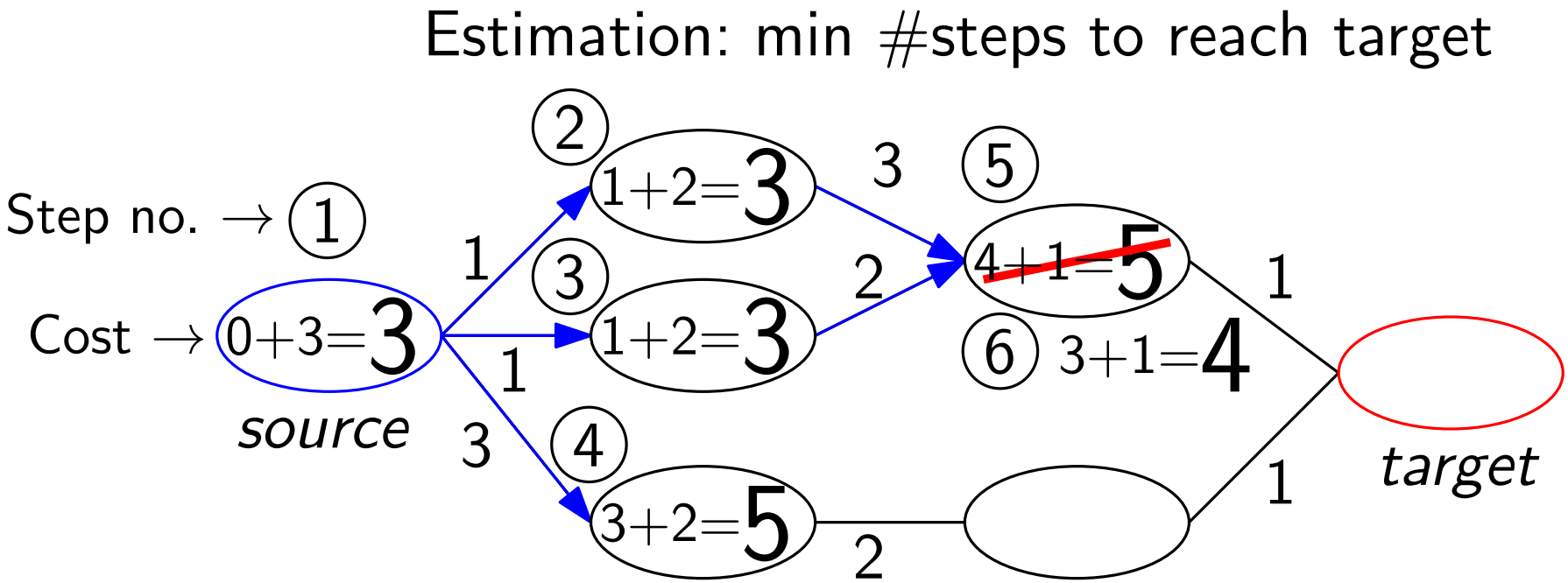
A* Algorithm



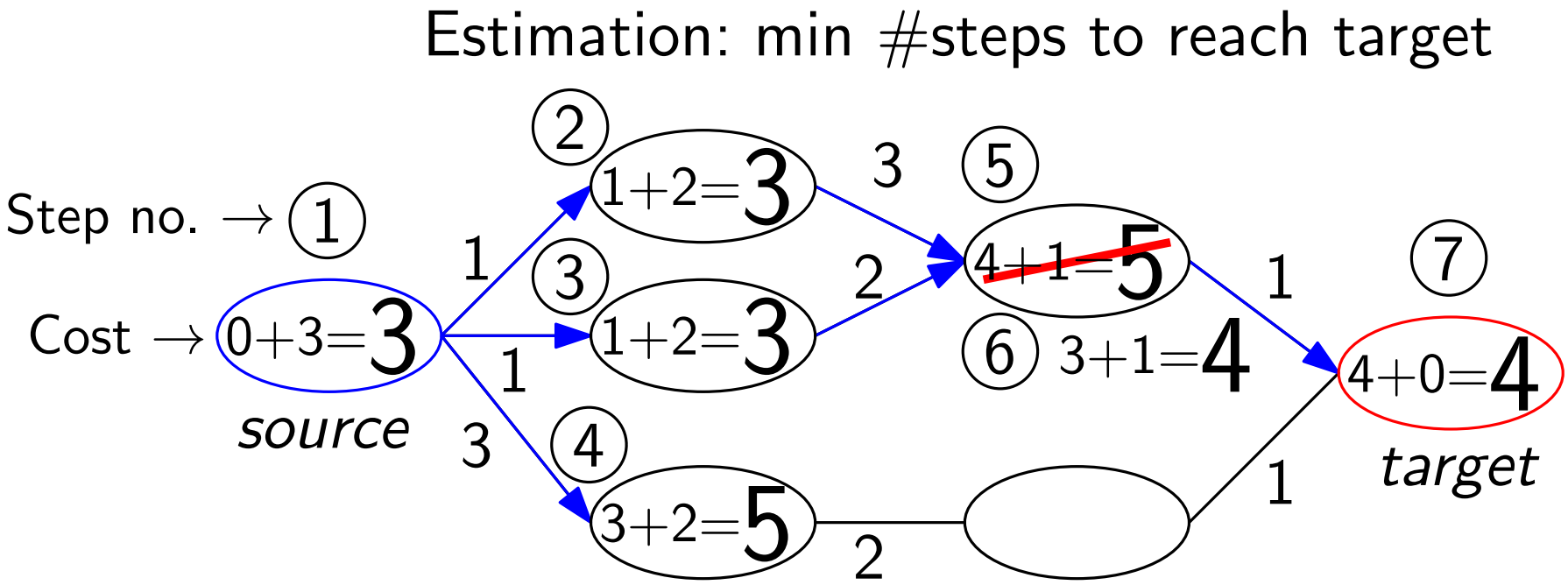
A* Algorithm



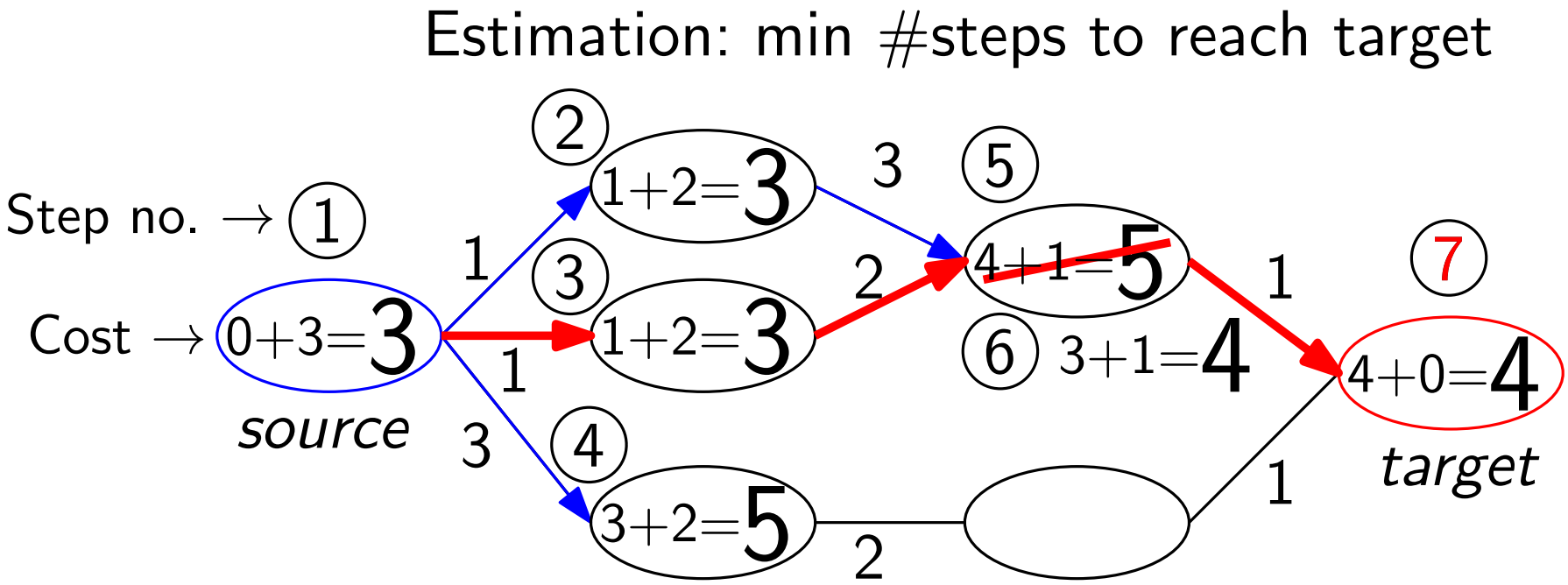
A* Algorithm



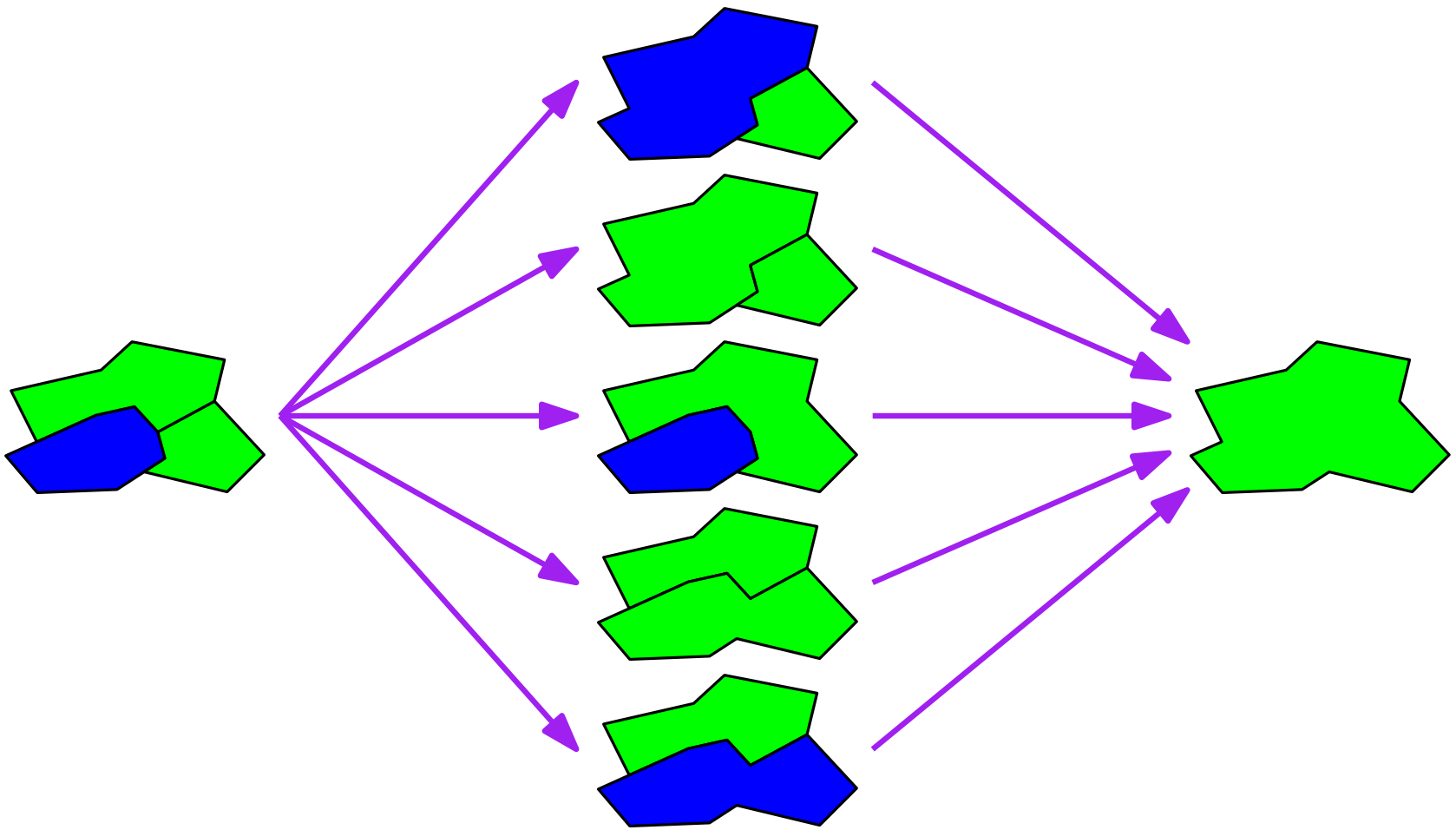
A* Algorithm


Estimation: min #steps to reach target
Step no. → 1
Cost → 0+3=3
source
2
1+2=3
3
1+2=3
4
3+2=5
5
4+1=5
6
3+1=4
7
4+0=4
target
1
1
3
3
2
2
1
1

A* Algorithm

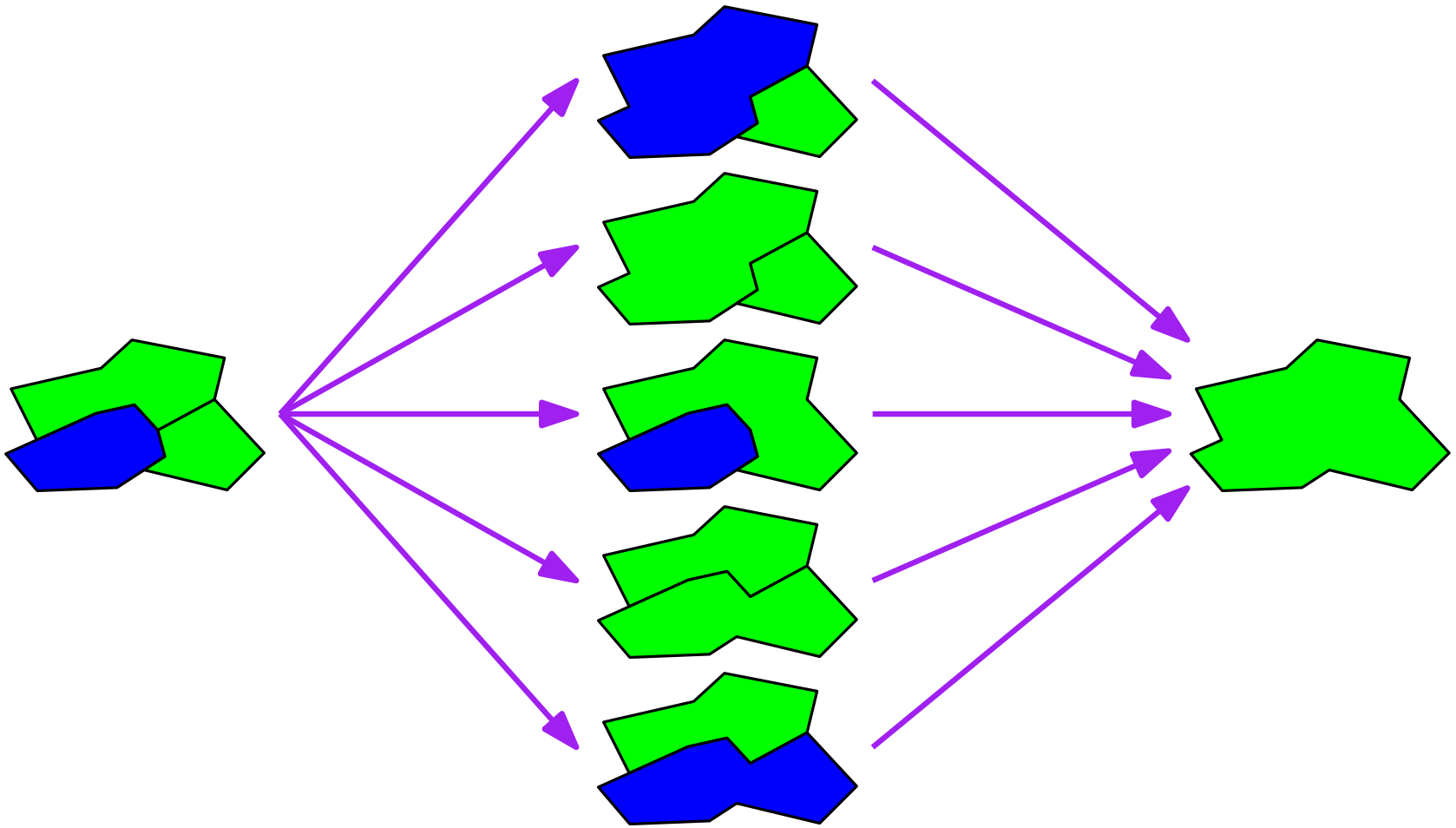


Formalizing a Pathfinding



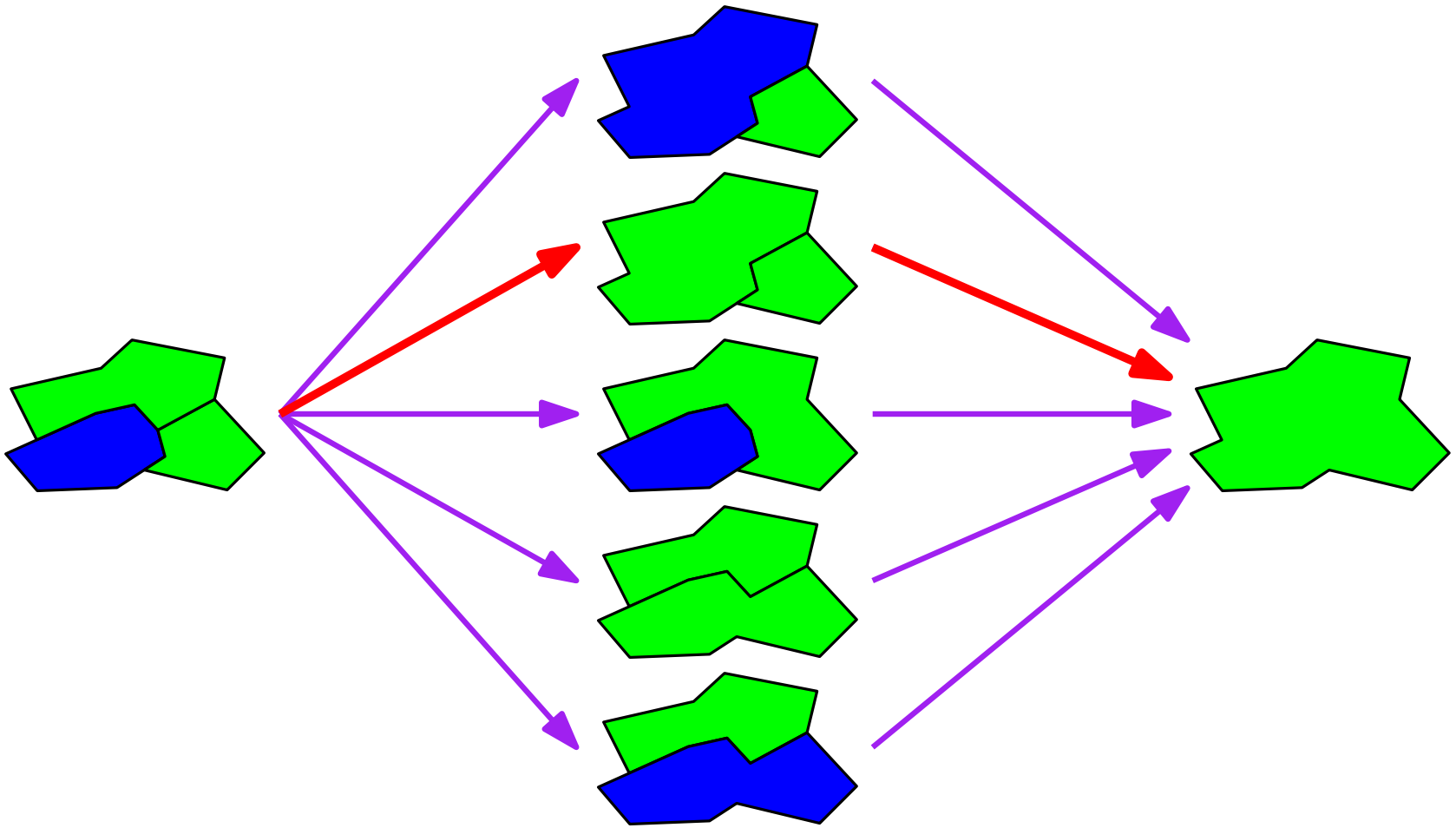
Formalizing a Pathfinding

- Each subdivision is represented as a node



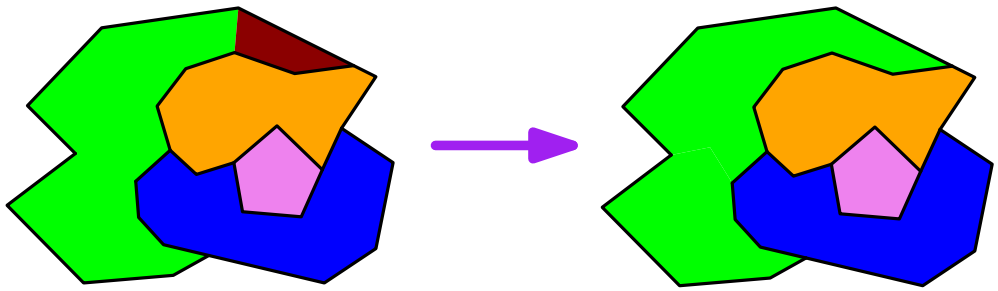
Formalizing a Pathfinding

- Each subdivision is represented as a node
- Find a shortest path with respect to cost functions



Cost Function

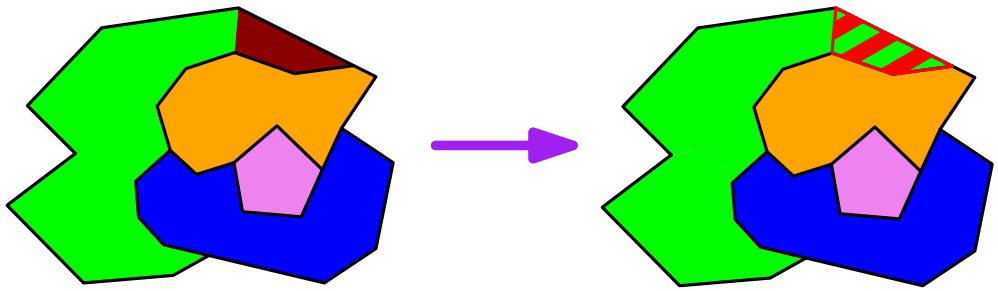
- Type change
  $f_{\text{type}}(\mathcal{P}_{s-1,i}, \mathcal{P}_{s,j}) =$



Cost Function

- Type change

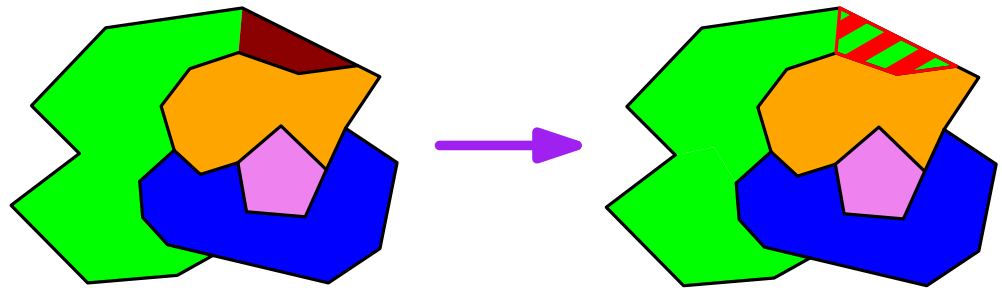
$$f_{\text{type}}(\mathcal{P}_{s-1,i}, \mathcal{P}_{s,j}) = \frac{A_p}{A_R} \cdot \frac{Cost(T(p), T(q))}{T_{\max}}$$



Cost Function

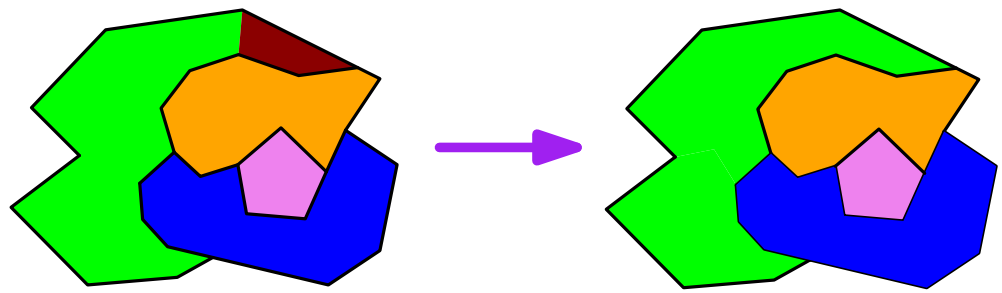
- Type change

$$f_{\text{type}}(\mathcal{P}_{s-1,i}, \mathcal{P}_{s,j}) = \frac{A_p}{A_R} \cdot \frac{Cost(T(p), T(q))}{T_{\max}}$$



- Shape (non-compactness)

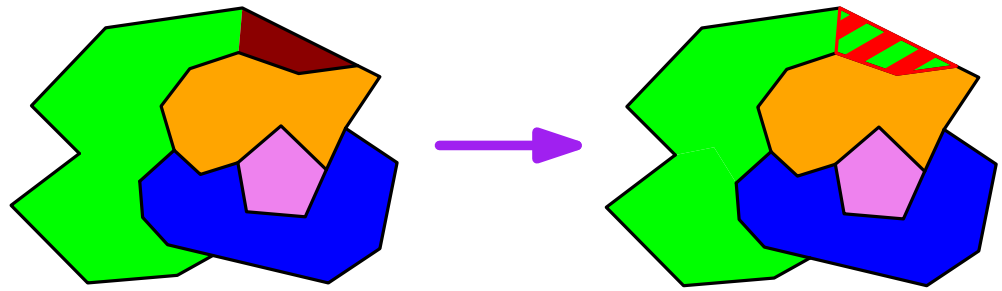
$$f_{\text{shape}}(\mathcal{P}_{s,k}) =$$



Cost Function

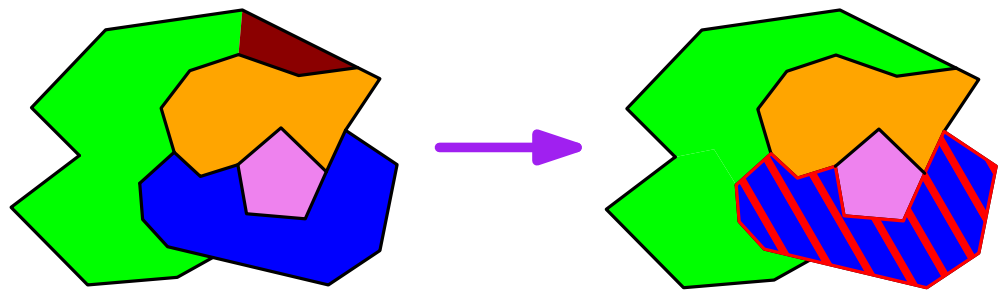
- Type change

$$f_{\text{type}}(\mathcal{P}_{s-1,i}, \mathcal{P}_{s,j}) = \frac{A_p}{A_R} \cdot \frac{Cost(T(p), T(q))}{T_{\max}}$$

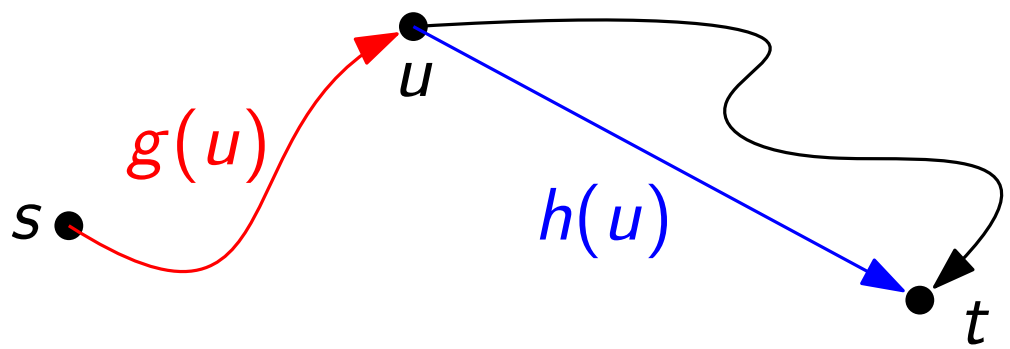


- Shape (non-compactness)

$$f_{\text{shape}}(\mathcal{P}_{s,k}) = \frac{1 - \min_{p \in \mathcal{P}_{s,k}} c_p}{n-2}$$

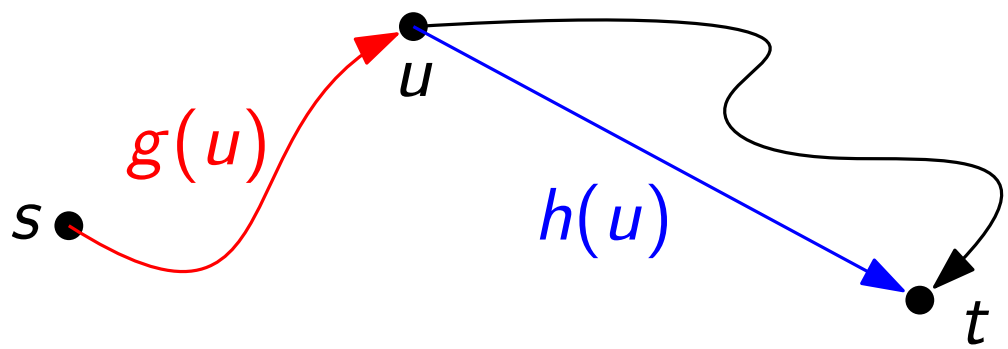


Exact Cost and Estimated Cost



Exact Cost and Estimated Cost

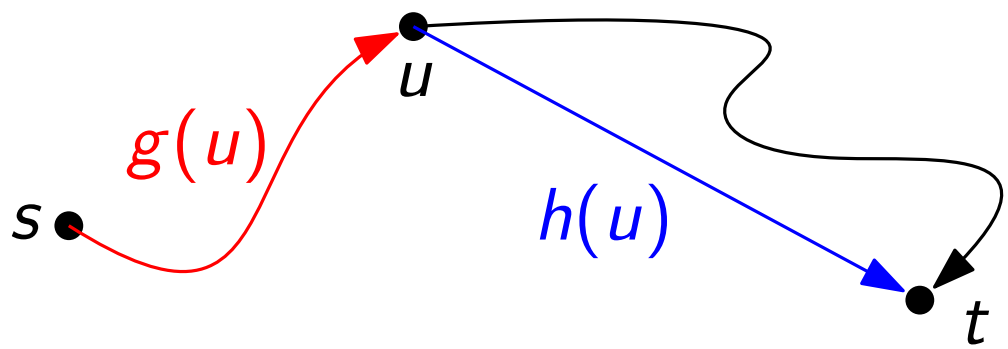
- *Exact cost*
  $g(\mathcal{P}_{t,i}) = (1-\lambda)\sum_{j=2}^{t} f_{\text{type}}(\mathcal{P}_{j-1,i_{j-1}}, \mathcal{P}_{j,i_j}) + \lambda \sum_{j=2}^{t-1} f_{\text{shape}}(\mathcal{P}_{j,i_j})$



Exact Cost and Estimated Cost

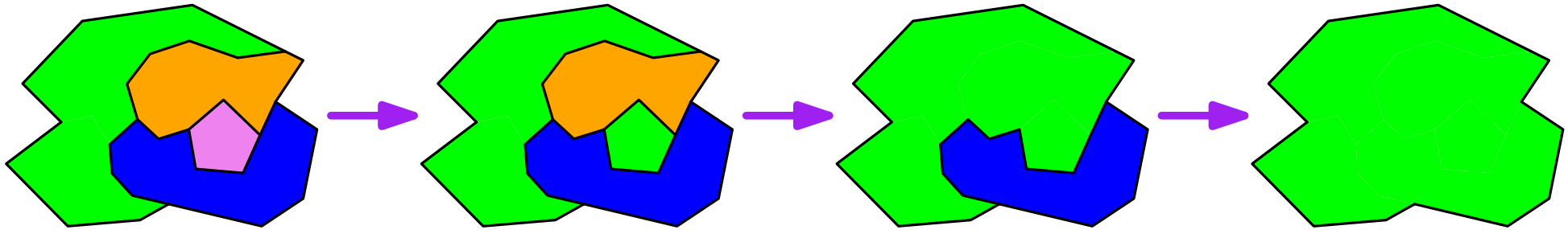
- *Exact cost*
  $g(\mathcal{P}_{t,i}) = (1-\lambda)\sum_{j=2}^{t} f_{\text{type}}(\mathcal{P}_{j-1,i_{j-1}}, \mathcal{P}_{j,i_j}) + \lambda \sum_{j=2}^{t-1} f_{\text{shape}}(\mathcal{P}_{j,i_j})$

- Estimated cost
  $h(\mathcal{P}_{t,i}) = (1-\lambda) h_{\text{type}}(\mathcal{P}_{t,i}) + \lambda h_{\text{shape}}(\mathcal{P}_{t,i})$



Estimated Cost

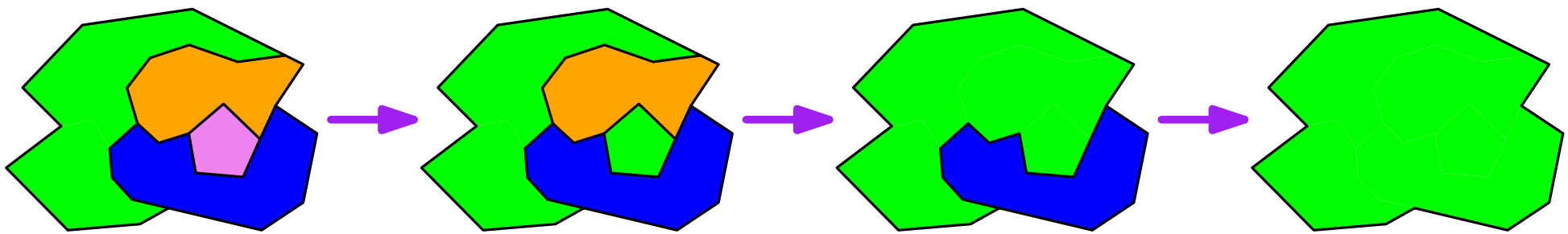
- $h_{\text{type}}(\mathcal{P}_{t,i}) = \sum_{s=t}^{n-1} f_{\text{type}}(\mathcal{P}_{s,i_s}, \mathcal{P}_{s+1,i_{s+1}})$.
  We assume: Each patch immediately gets the target type.



$t = 2$ $t = 3$ $t = 4$ $t = 5$

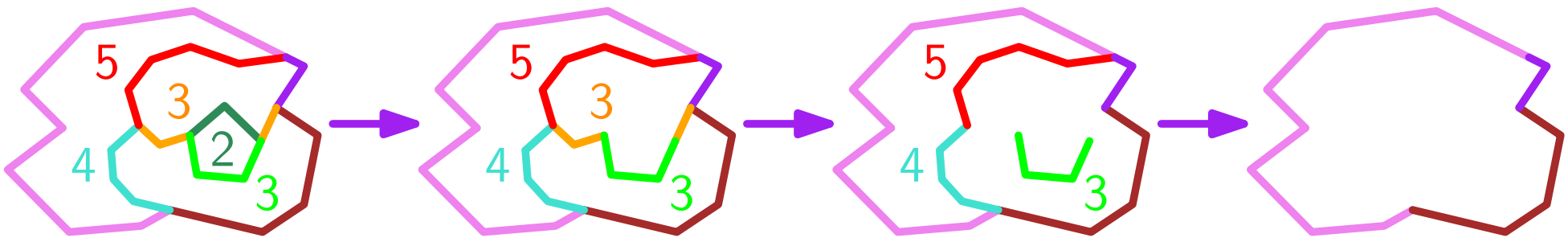
Estimated Cost

- $h_{\text{type}}(\mathcal{P}_{t,i}) = \sum_{s=t}^{n-1} f_{\text{type}}(\mathcal{P}_{s,i_s}, \mathcal{P}_{s+1,i_{s+1}})$.
  We assume: Each patch immediately gets the target type.



$t = 2$ $t = 3$ $t = 4$ $t = 5$

- $h_{\text{shape}}(\mathcal{P}_{t,i}) = \sum_{s=t}^{n-1} C(\mathcal{P}_{s,j_s})$,
  The more the edges remain, the smaller the estimated compactness $C(\mathcal{P}_{s,j_s})$ is
  We assume: The boundary with the fewest edges is removed.



$n_{\text{int}}(\mathcal{P}_{2,j_2}) = 17$ $n_{\text{int}}(\mathcal{P}_{3,j_3}) = 15$ $n_{\text{int}}(\mathcal{P}_{4,j_4}) = 12$ $n_{\text{int}}(\mathcal{P}_{5,j_5}) = 0$

Overestimation

- Try finding a path by exploring M nodes. If we do not find one, then we overestimate and try exploring M nodes again.
- Overestimate by increasing $k \in \{0, 1, \dots\}$:
  $F(\mathcal{P}_{t,i}) = g(\mathcal{P}_{t,i}) + 2^k h(\mathcal{P}_{t,i})$

Overestimation

- Try finding a path by exploring M nodes. If we do not find one, then we overestimate and try exploring M nodes again.
- Overestimate by increasing $k \in \{0, 1, \dots\}$:
  $F(\mathcal{P}_{t,i}) = g(\mathcal{P}_{t,i}) + 2^k h(\mathcal{P}_{t,i})$

  With a larger k, a path seems more expensive (or longer), thus may not be explored

Overestimation

- Try finding a path by exploring M nodes. If we do not find one, then we overestimate and try exploring M nodes again.
- Overestimate by increasing $k \in \{0, 1, \dots\}$:
  $F(\mathcal{P}_{t,i}) = g(\mathcal{P}_{t,i}) + 2^k h(\mathcal{P}_{t,i})$

  With a larger k, a path seems more expensive (or longer), thus may not be explored

Not optimal anymore if $k > 0$!

Outline

- Introduction
- Methodology
- Case Study
- Concluding Remarks

Case Study

Environment

- C# (using the .NET Framework 4.0)
- ArcObjects SDK 10.2
- Windows 7, 3.3 GHz dual core CPU, 8 GB RAM
- Time measure: Stopwatch (a class in C#)

Case Study

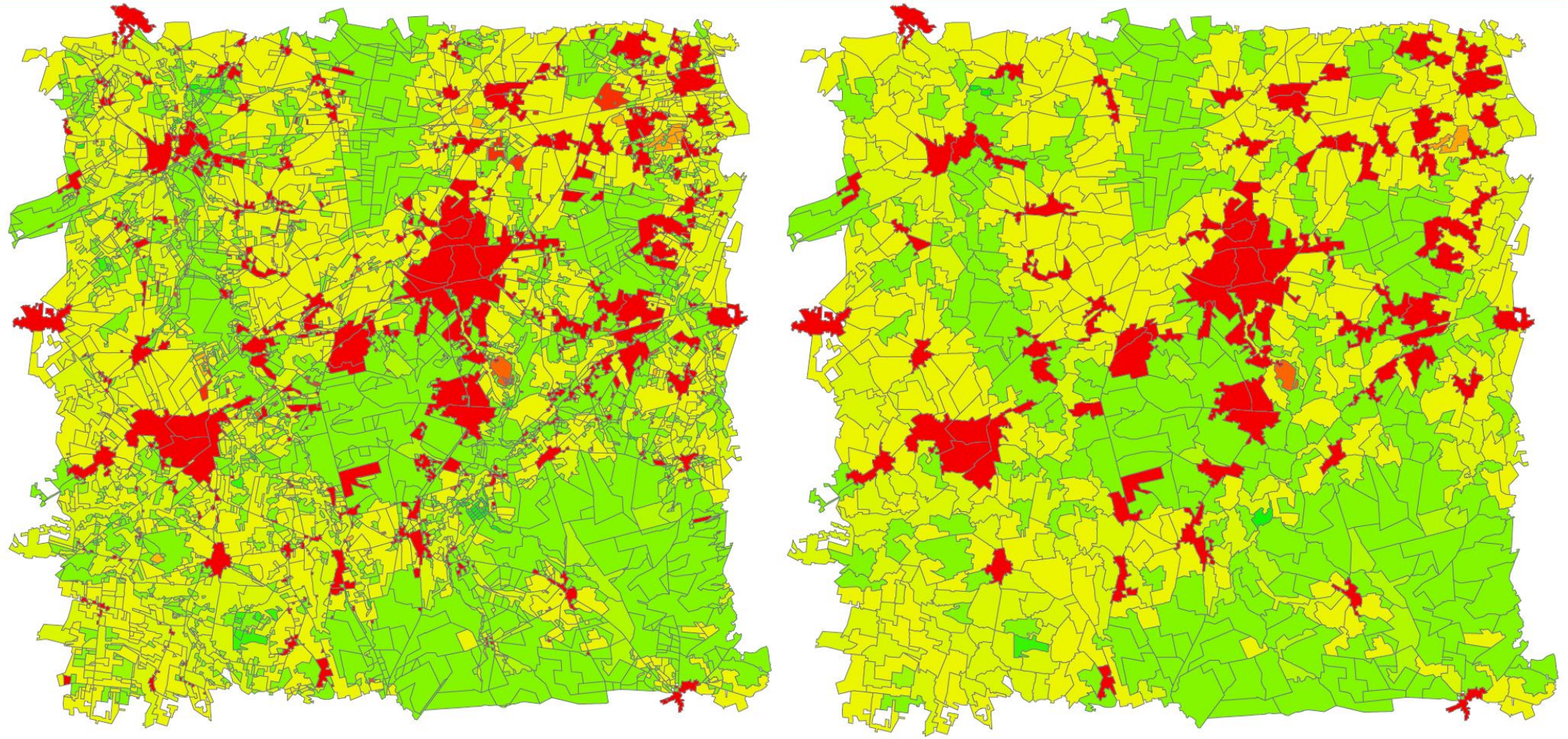
Environment

- C# (using the .NET Framework 4.0)
- ArcObjects SDK 10.2
- Windows 7, 3.3 GHz dual core CPU, 8 GB RAM
- Time measure: Stopwatch (a class in C#)

Data

- Source: scale 1 : 50,000; 5,448 patches; place "Buchholz in der Nordheide"; from ATKIS (Das Amtliche Topographisch-Kartographische Informationssystem), processed by Haunert (2009, pp. 61–66)
- Target: scale 1 : 250,000; 730 patches; generalized from the source by Haunert and Wolff (2010)

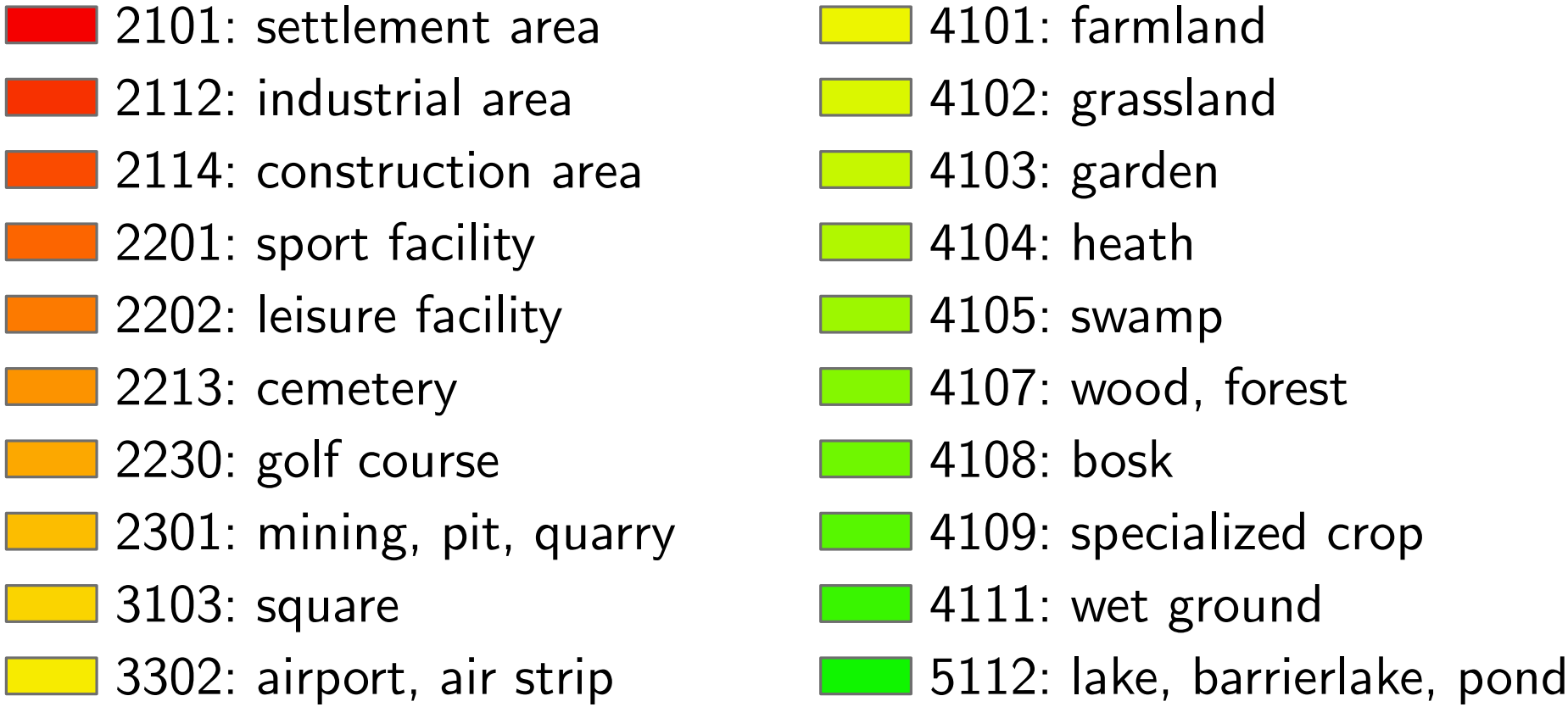
Data



5,448 patches, scale 1 : 50,000

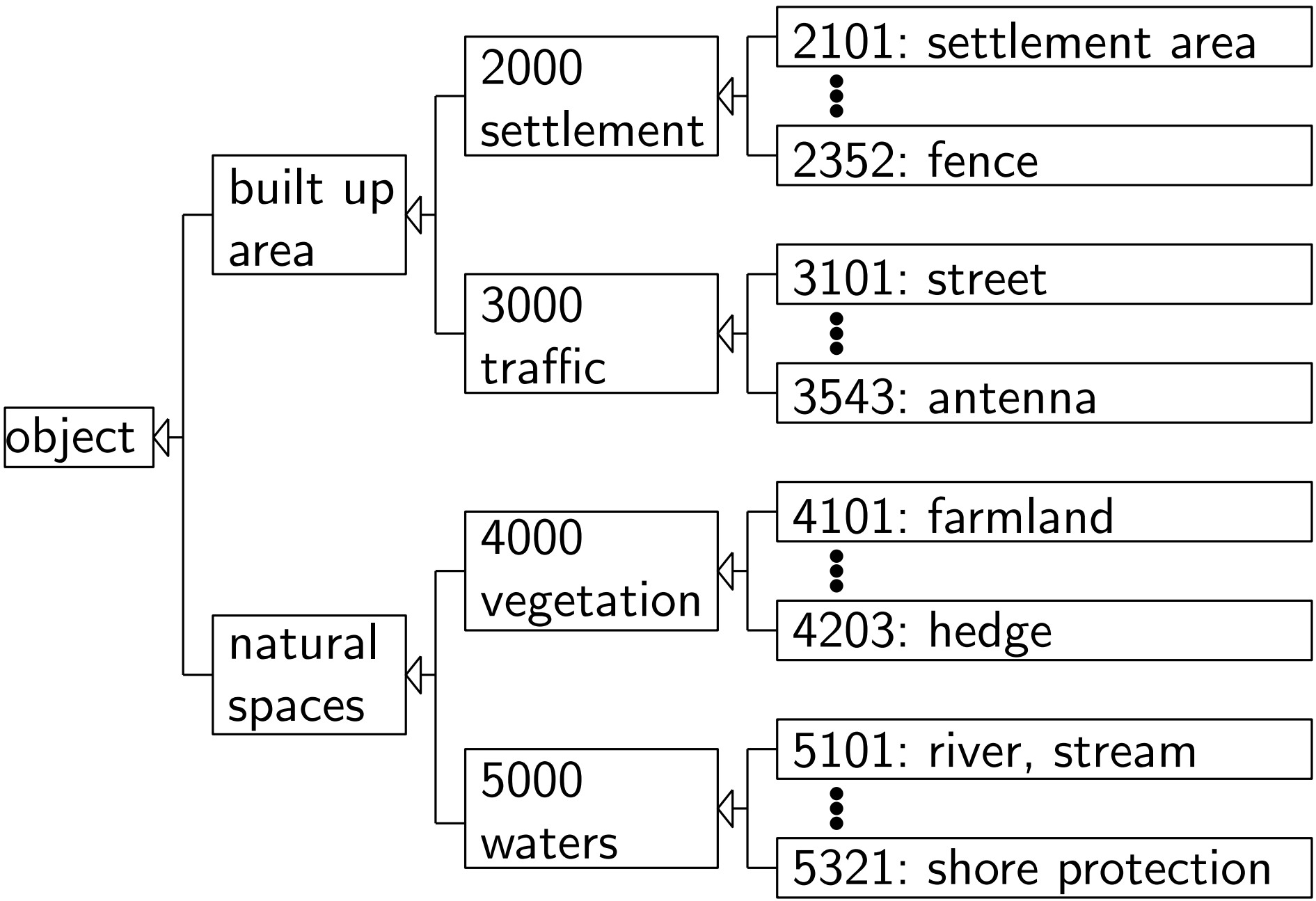
730 patches, scale 1 : 250,000

Legend



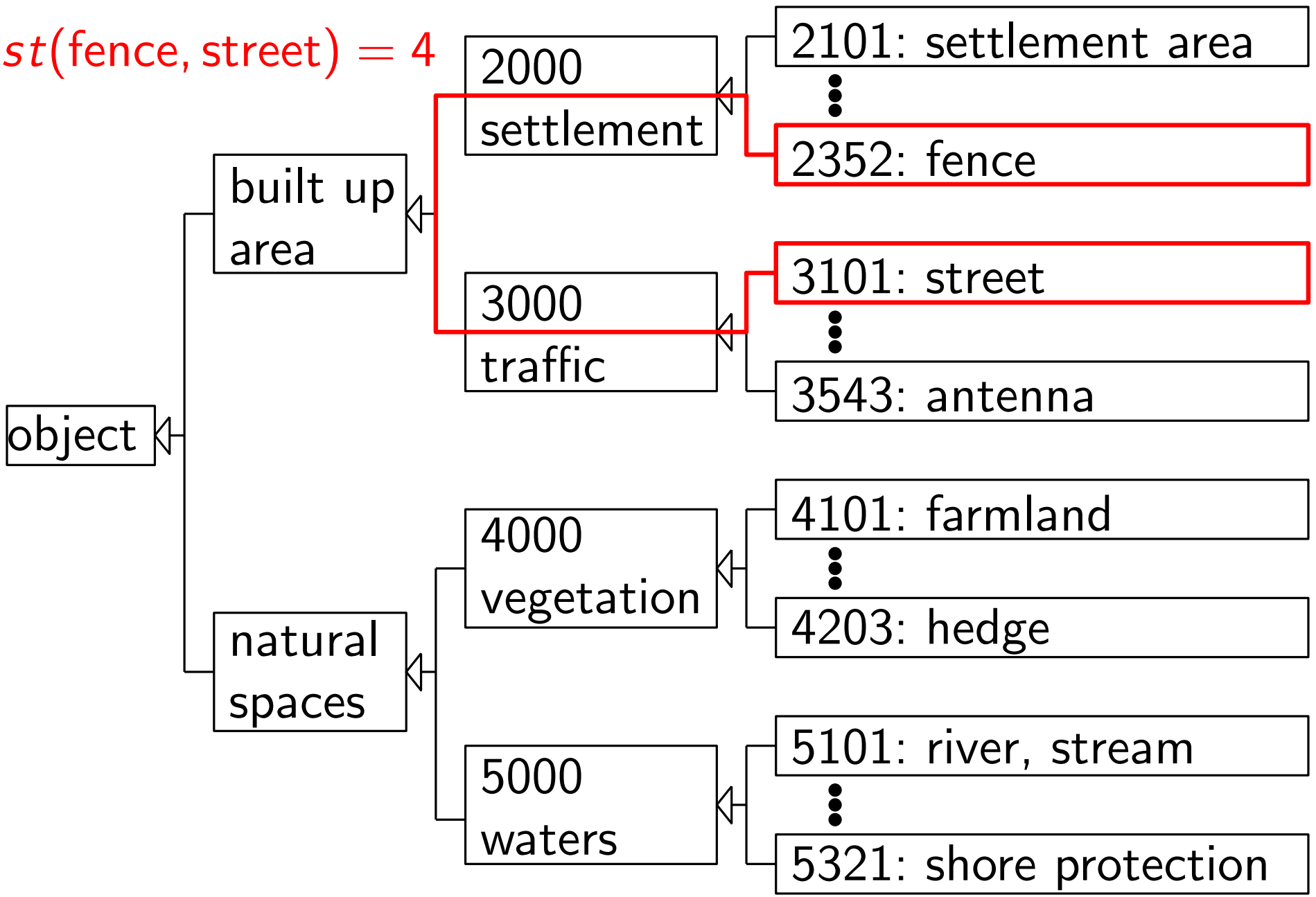
The 20 land-cover types appearing in our data

Type Distance



Type Distance

$Cost(\text{fence}, \text{street}) = 4$



Result

- $\lambda = 0.5,\ M = 200{,}000$

Result

- $\lambda = 0.5,\ M = 200{,}000$
- Found optimal solutions for 692 regions out of 734

Result

- $\lambda = 0.5,\ M = 200{,}000$
- Found optimal solutions for 692 regions out of 734
- Explored $4.9 \cdot 10^6$ arcs and $2.4 \cdot 10^6$ nodes

Result

- $\lambda = 0.5,\ M = 200{,}000$
- Found optimal solutions for 692 regions out of 734
- Explored $4.9 \cdot 10^6$ arcs and $2.4 \cdot 10^6$ nodes
- Running time: 36 min

Result

- $\lambda = 0.5,\ M = 200{,}000$
- Found optimal solutions for 692 regions out of 734
- Explored $4.9 \cdot 10^6$ arcs and $2.4 \cdot 10^6$ nodes
- Running time: 36 min
- Largest overestimation factor: 512

Cost in detail

ID	n	m	K_i	R_{type}	R_{shape}	Time (s)
543	20	40	512	1.000	0.516	175.7
94	32	74	64	0.016	4.577	163.6
⋮	⋮	⋮	⋮	⋮	⋮	⋮
358	21	32	1	1.000	1020.133	1.3
97	20	35	1	1.000	179.132	0.9
257	20	33	1	1.000	463.178	5.9
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Cost in detail

ID	n	m	K_i	R_{type}	R_{shape}	Time (s)
543	20	40	512	1.000	0.516	175.7
94	32	74	64	0.016	4.577	163.6
⋮	⋮	⋮	⋮	⋮	⋮	⋮
358	21	32	1	1.000	1020.133	1.3
97	20	35	1	1.000	179.132	0.9
257	20	33	1	1.000	463.178	5.9
⋮	⋮	⋮	⋮	⋮	⋮	⋮

$R_{\text{type}} = g_{\text{type}}(\mathcal{P}_{\text{goal}})/h_{\text{type}}(\mathcal{P}_{\text{start}})$

Good estimation for type change

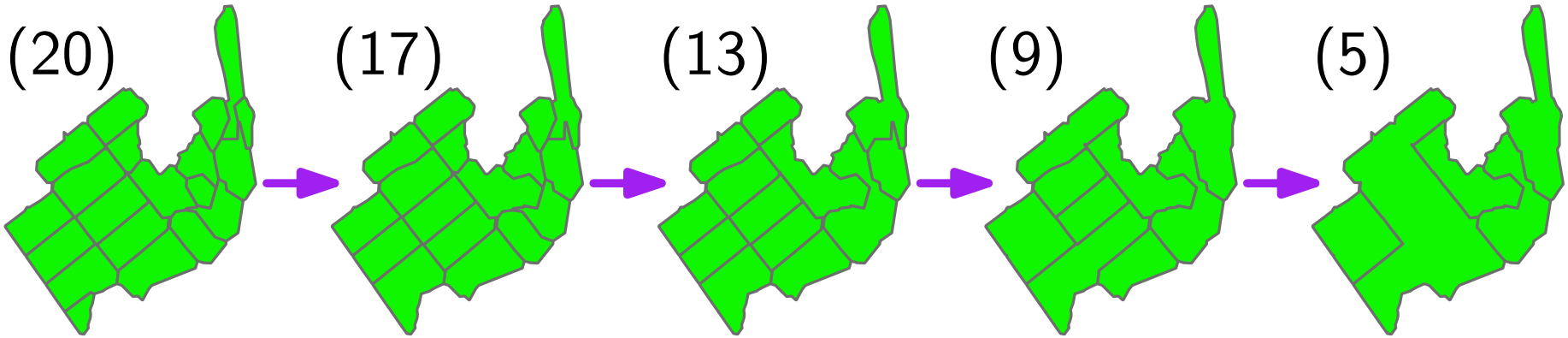
Cost in detail

ID	n	m	K_i	R_{type}	R_{shape}	Time (s)
543	20	40	512	1.000	0.516	175.7
94	32	74	64	0.016	4.577	163.6
⋮	⋮	⋮	⋮	⋮	⋮	⋮
358	21	32	1	1.000	1020.133	1.3
97	20	35	1	1.000	179.132	0.9
257	20	33	1	1.000	463.178	5.9
⋮	⋮	⋮	⋮	⋮	⋮	⋮

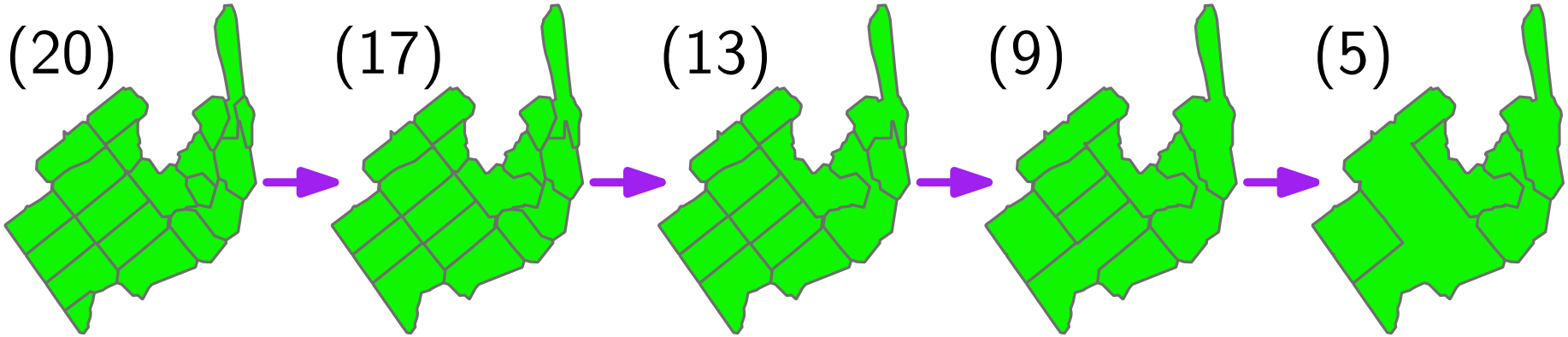
$R_{\text{type}} = g_{\text{type}}(\mathcal{P}_{\text{goal}})/h_{\text{type}}(\mathcal{P}_{\text{start}})$
Good estimation for type change

$R_{\text{shape}} = g_{\text{shape}}(\mathcal{P}_{\text{goal}})/h_{\text{shape}}(\mathcal{P}_{\text{start}})$
Poor estimation for shape (non-compactness)

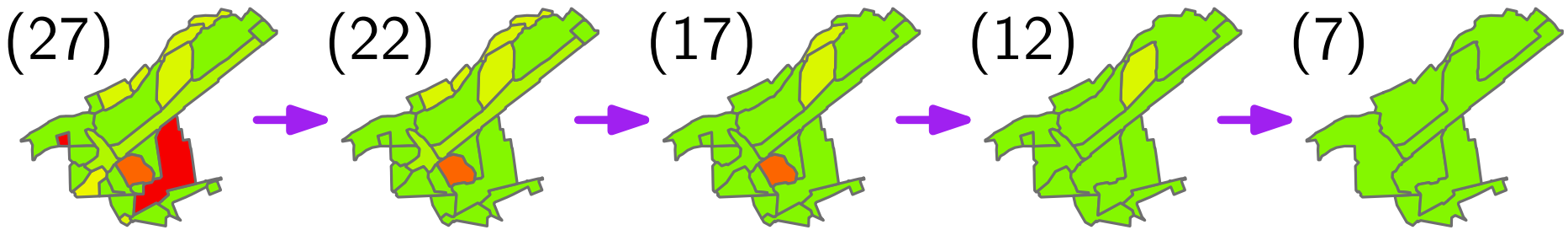
$K_{543} = 512$



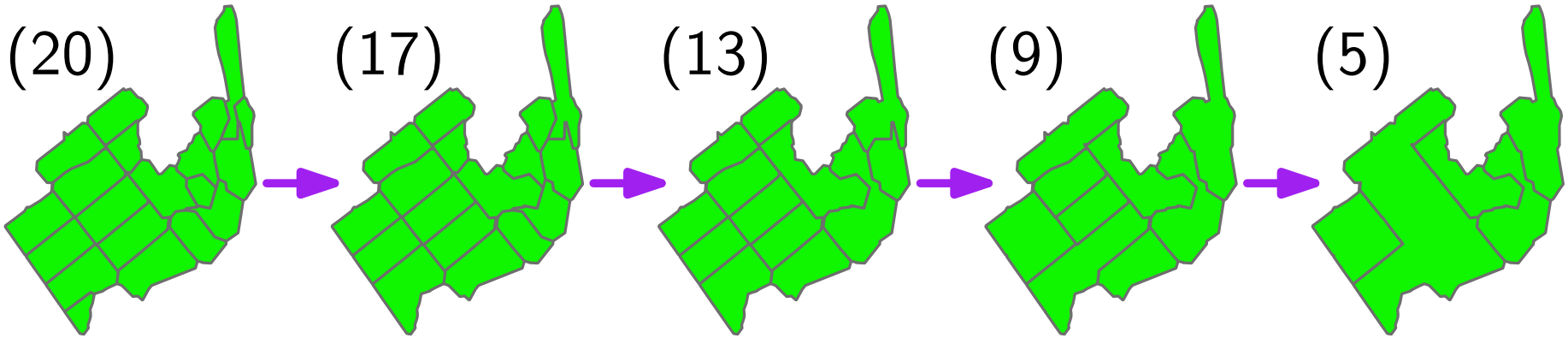
$K_{543} = 512$



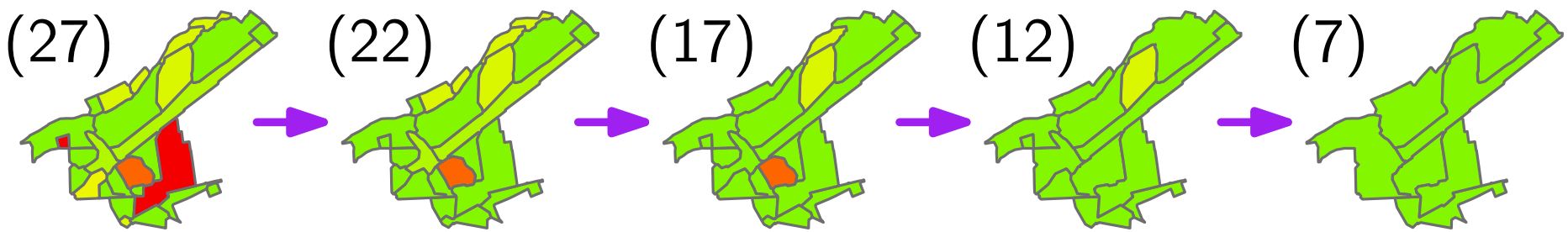
$K_{436} = 8$



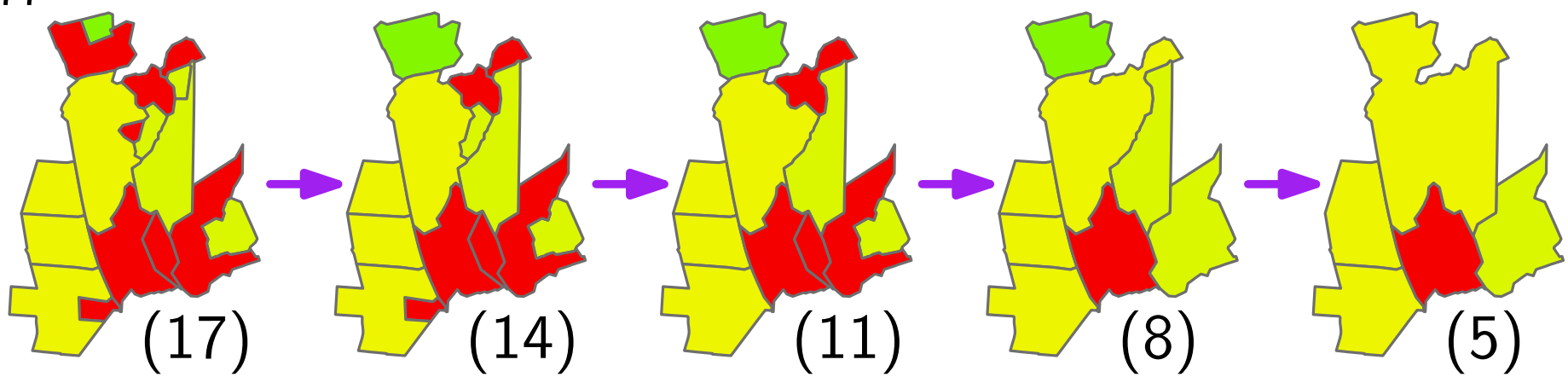
$K_{543} = 512$



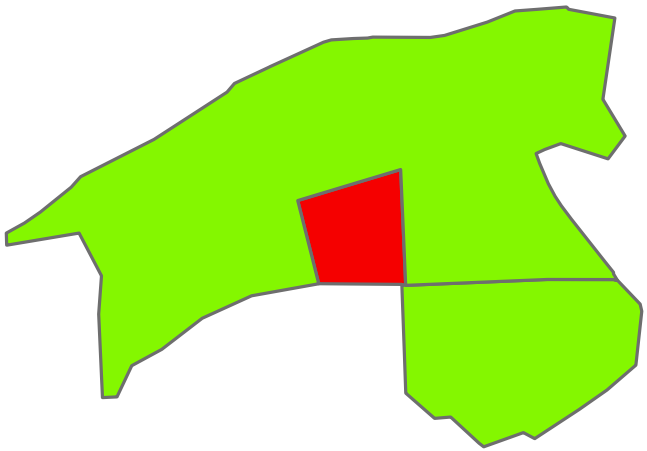
$K_{436} = 8$



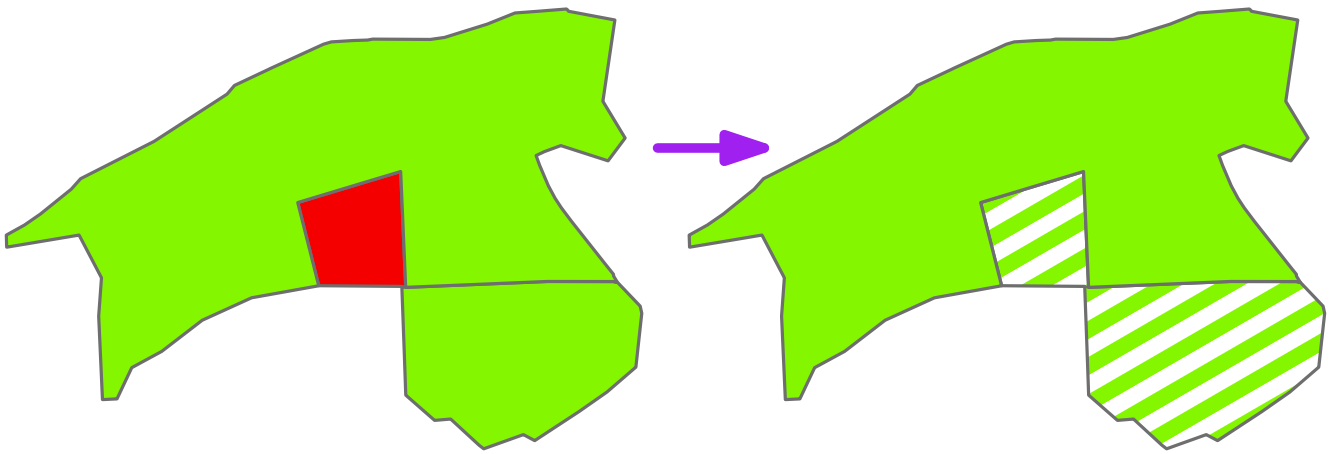
$K_{77} = 1$



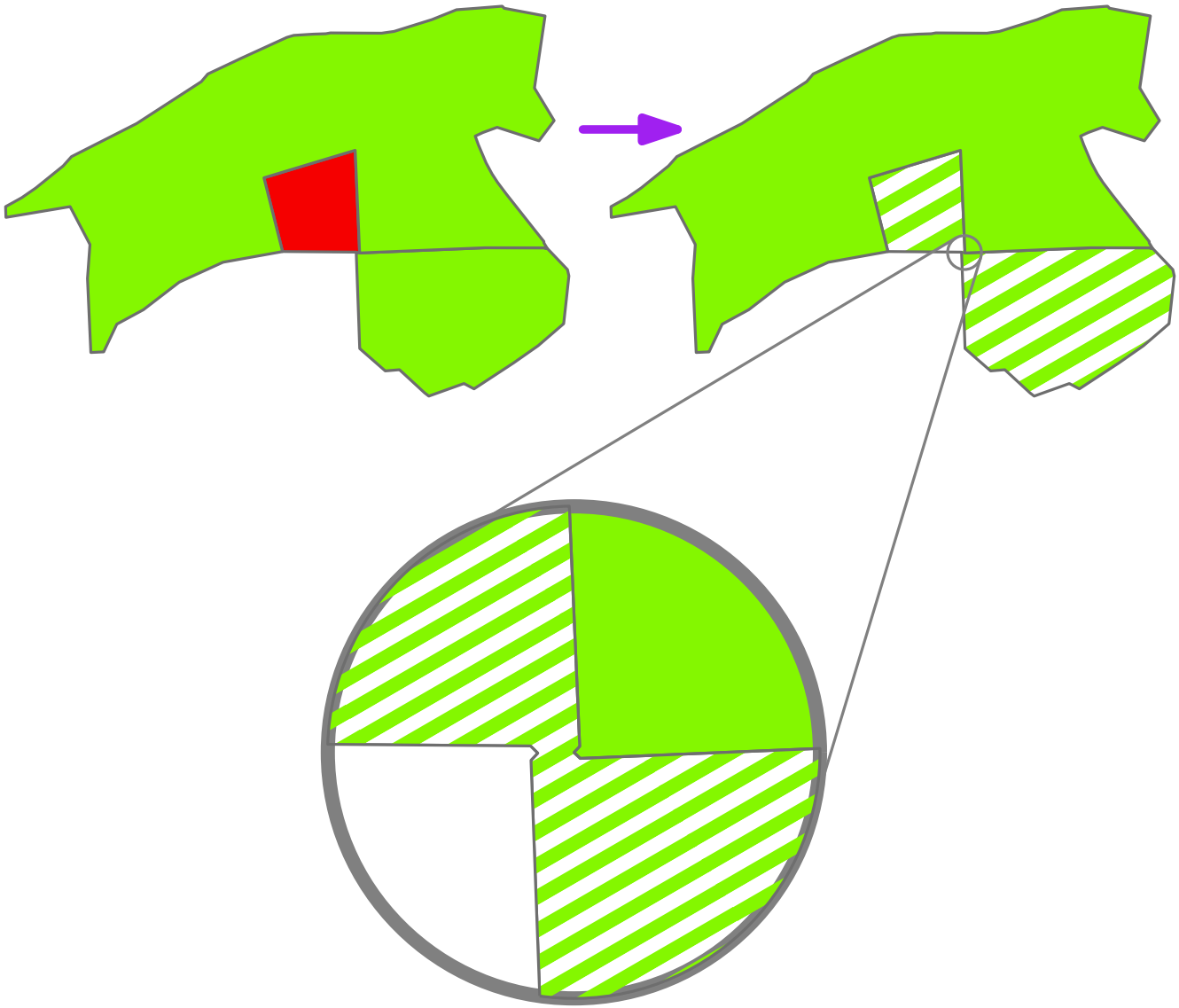
Inappropriate Step



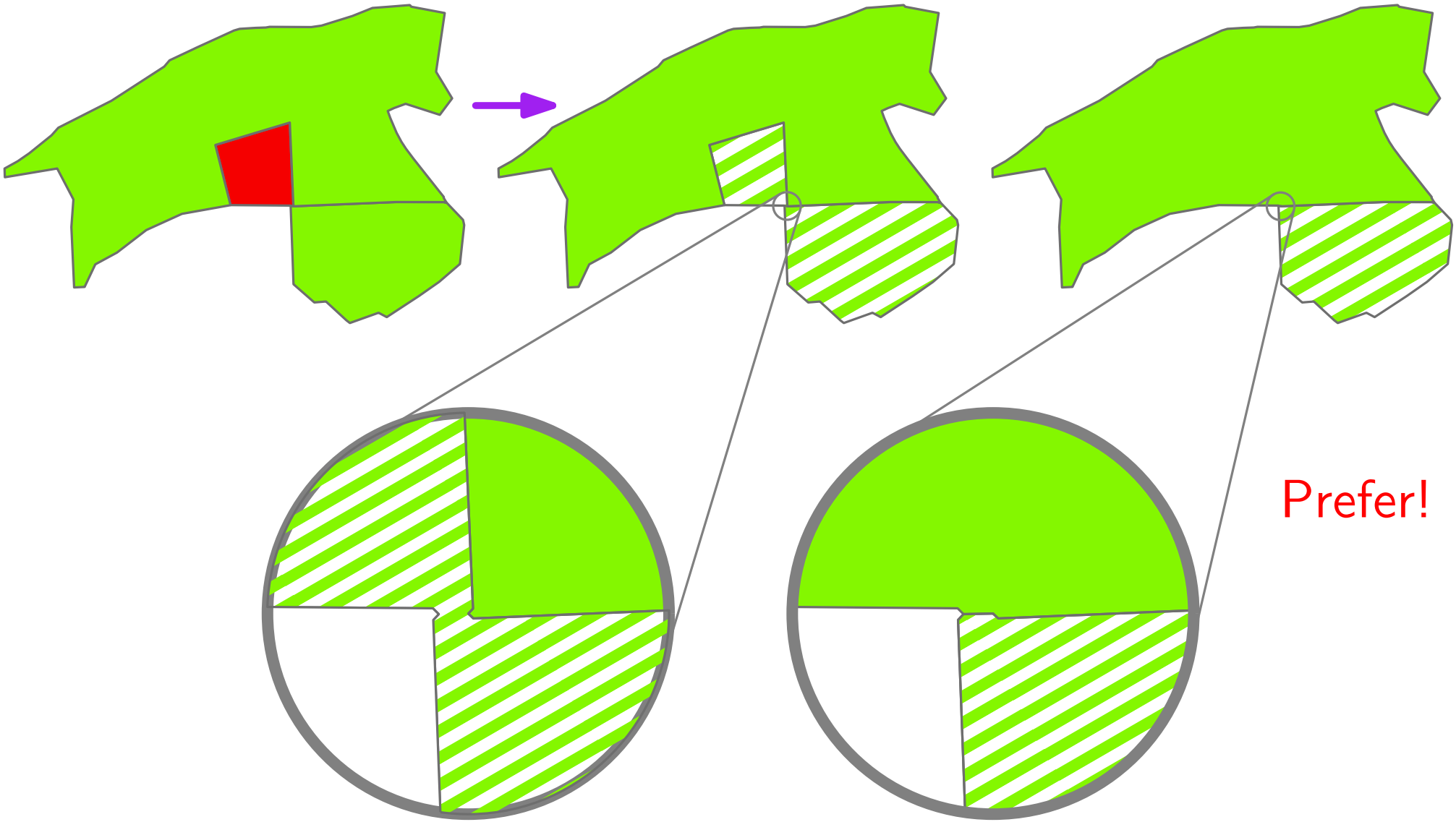
Inappropriate Step



Inappropriate Step



Inappropriate Step

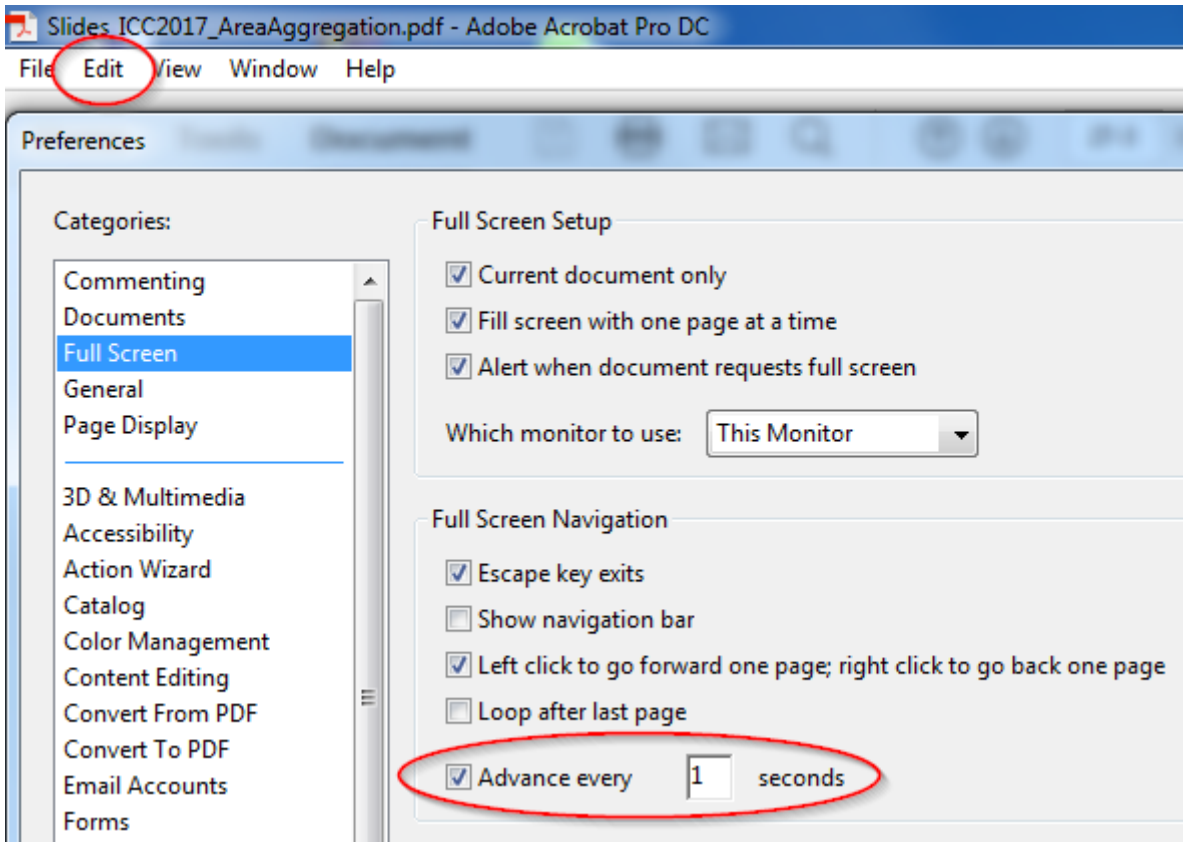


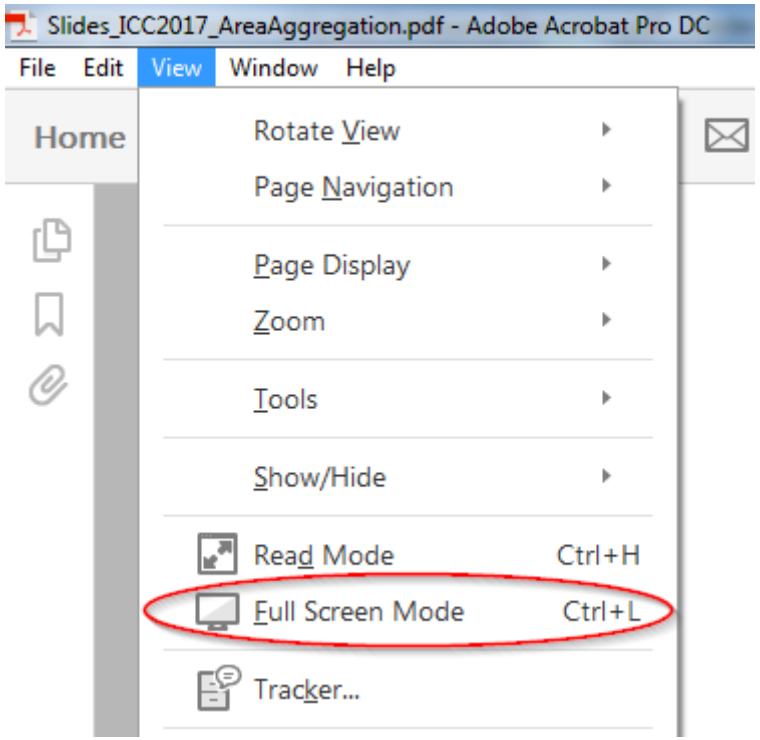
Animation

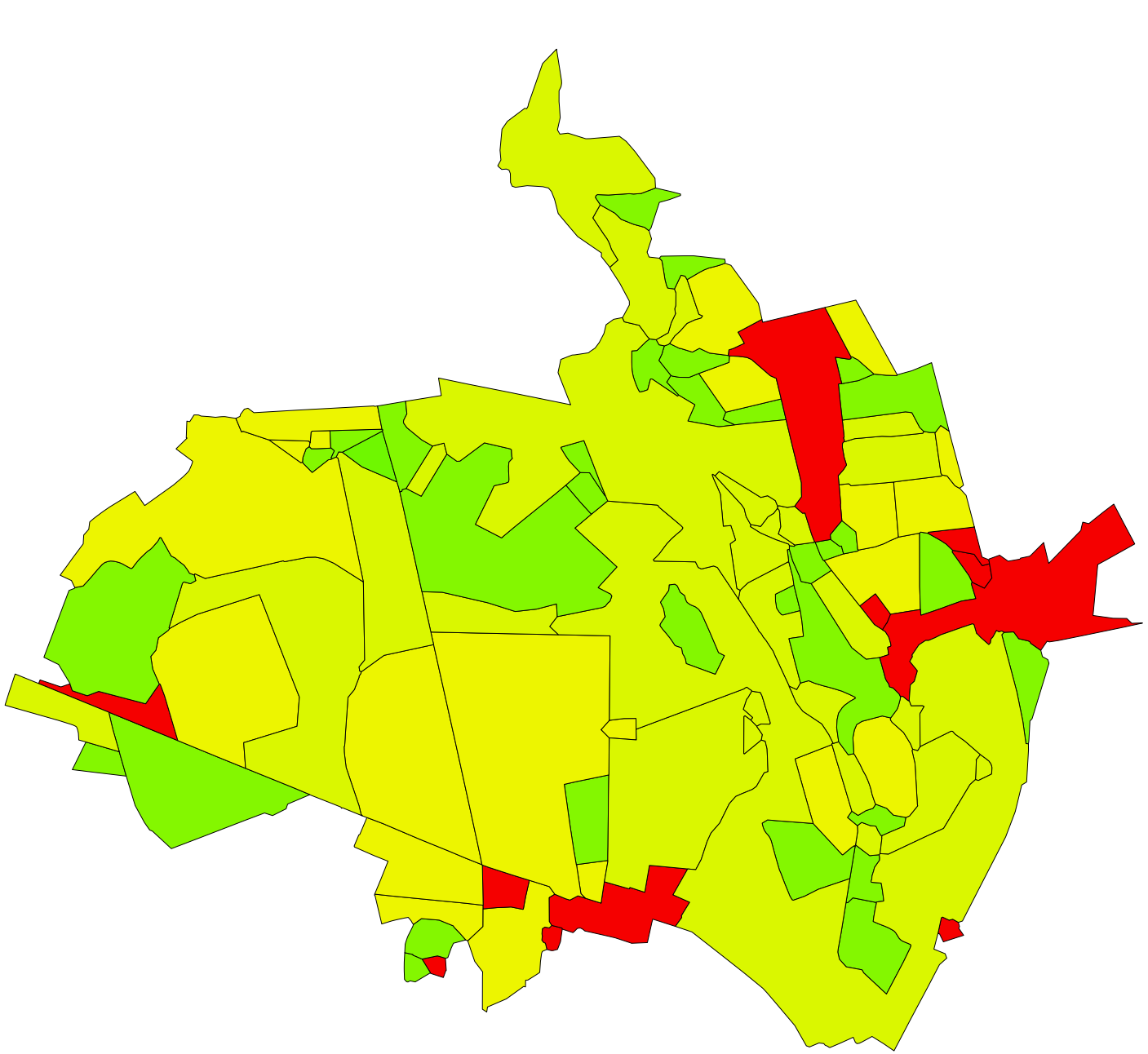
- **Thick blue polyline** represents a patch to be aggregated, and thin blue polyline represents an aggregated result.

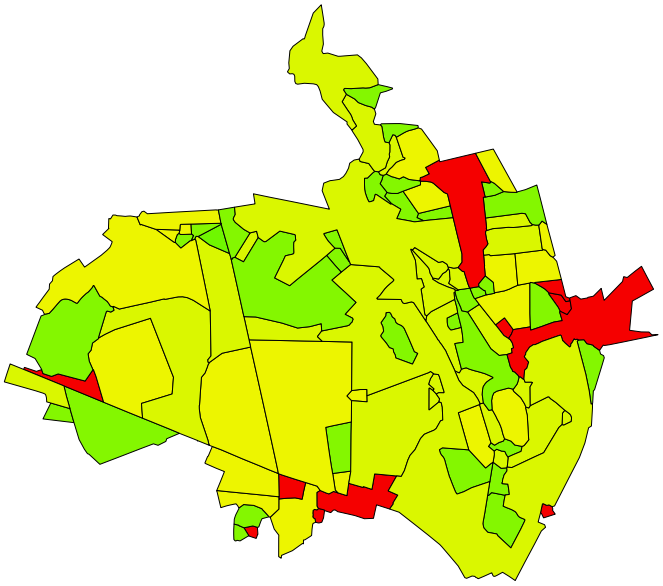
Animation

- **Thick blue polyline** represents a patch to be aggregated, and thin blue polyline represents an aggregated result.
- In Adobe Acrobat Pro, set "Advance every" to 1 second, then watch animation with "Full Screen Mode".

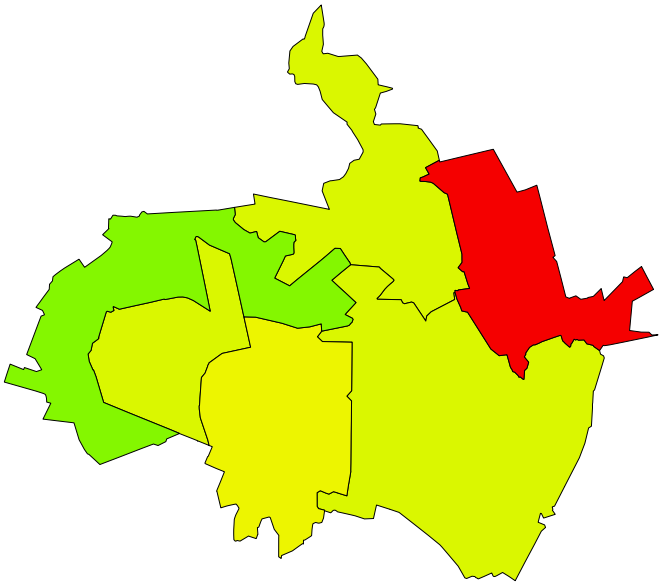




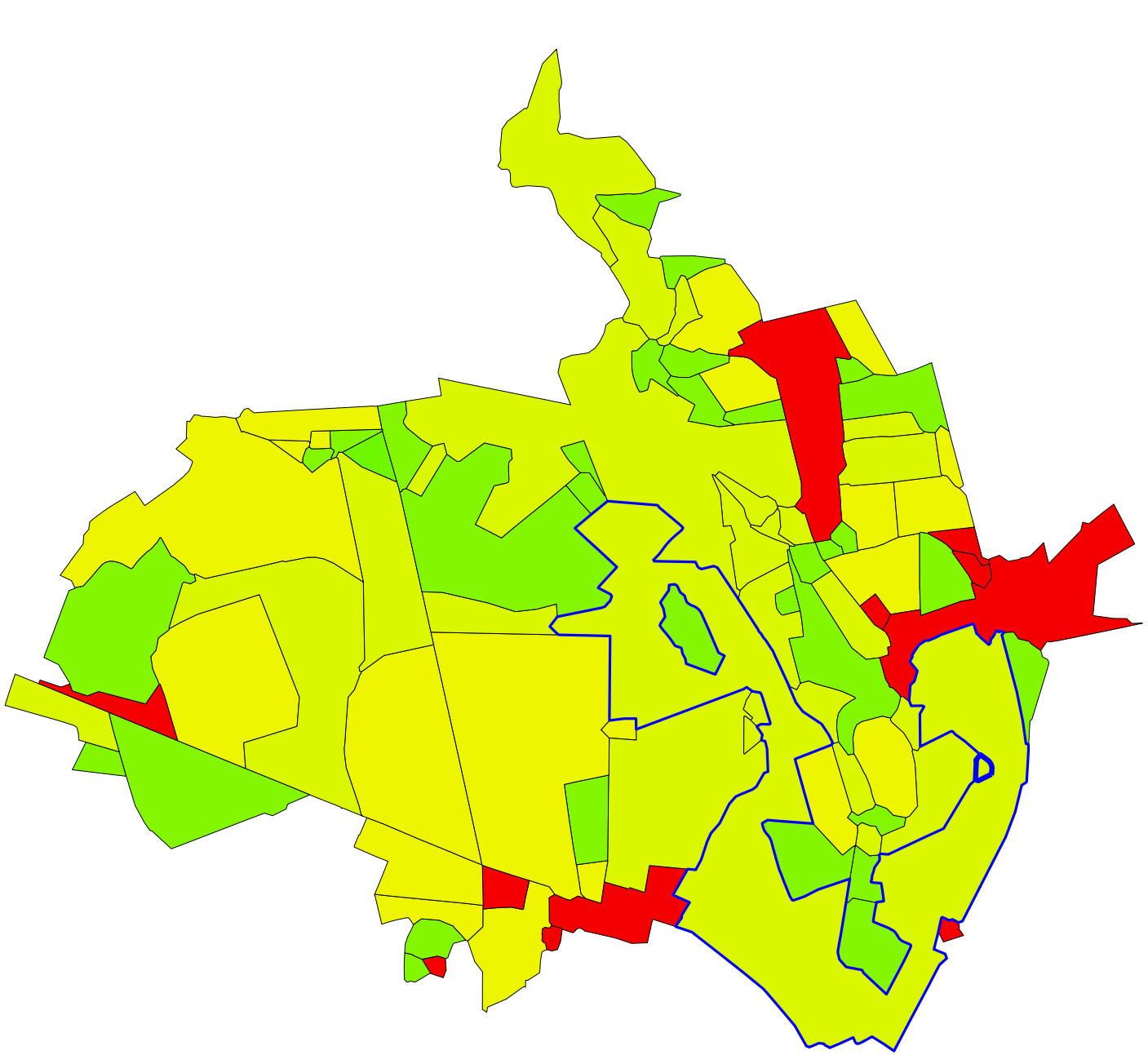


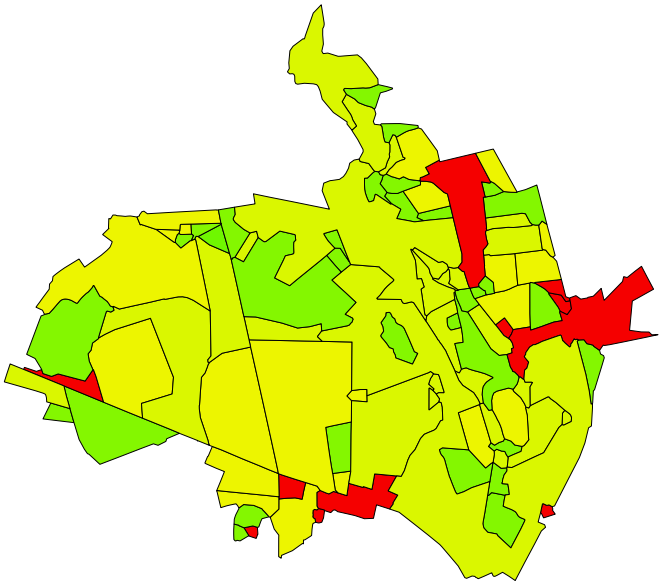


source: 92 patches

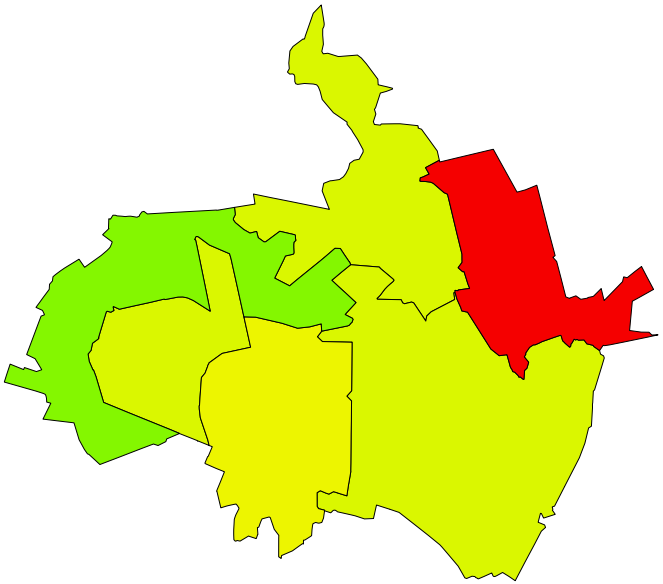


target: 6 patches

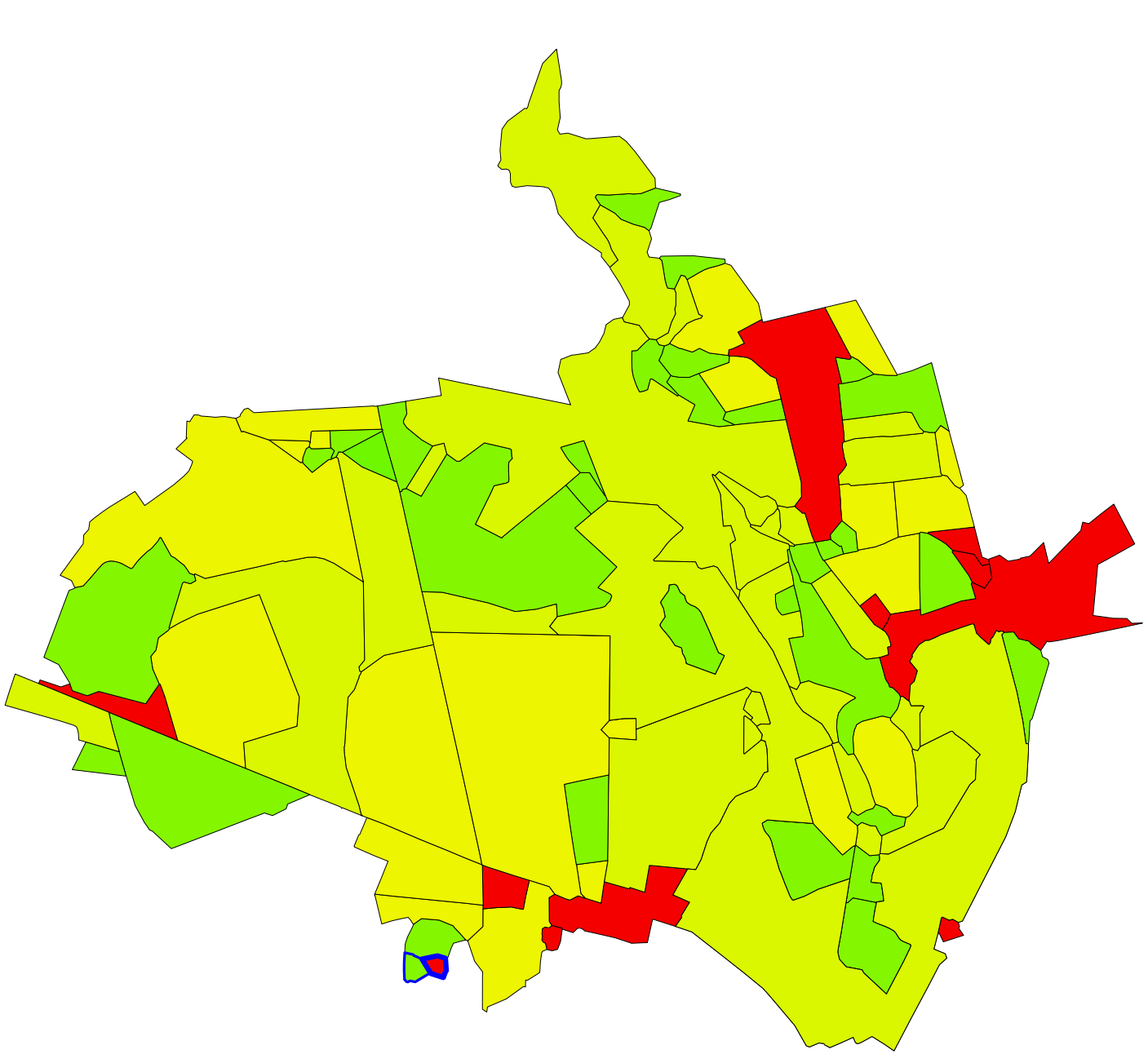


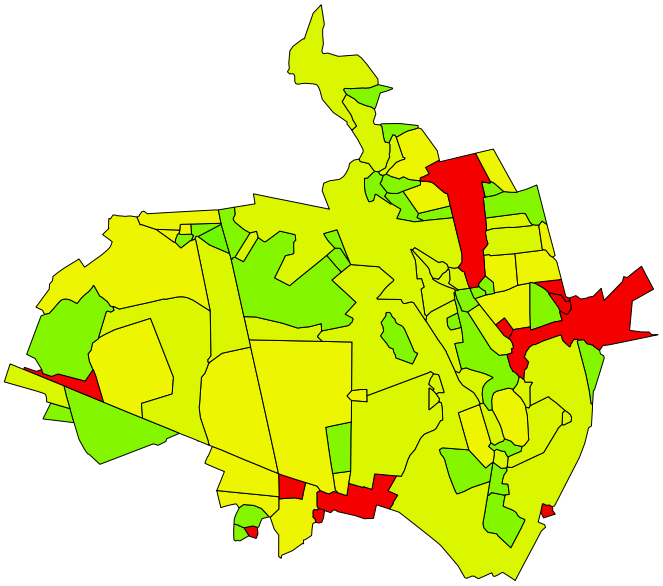


source: 92 patches

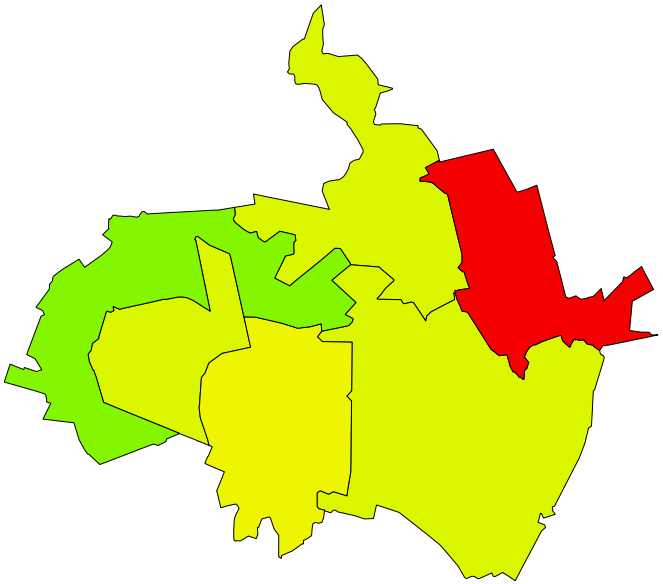


target: 6 patches

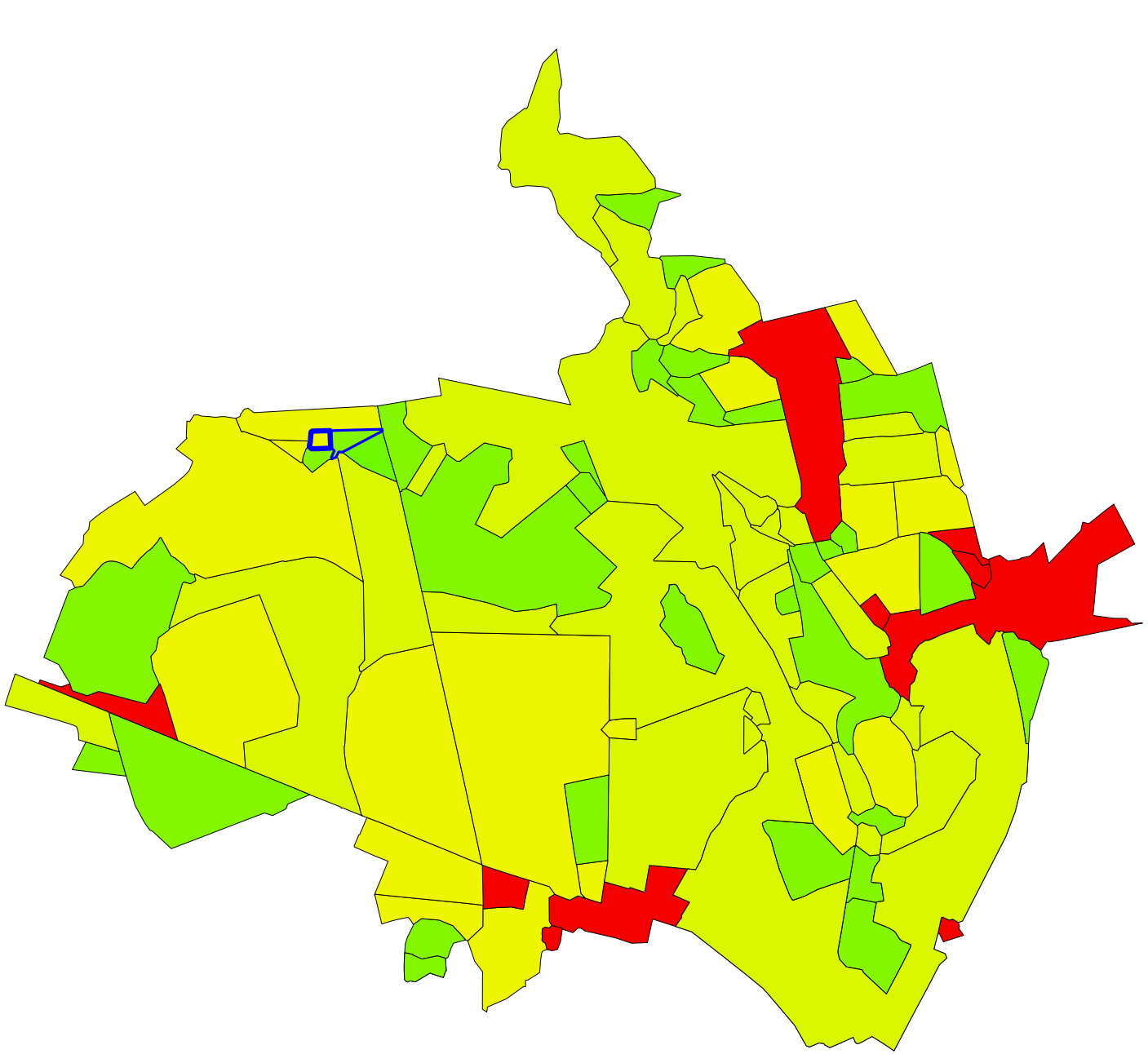


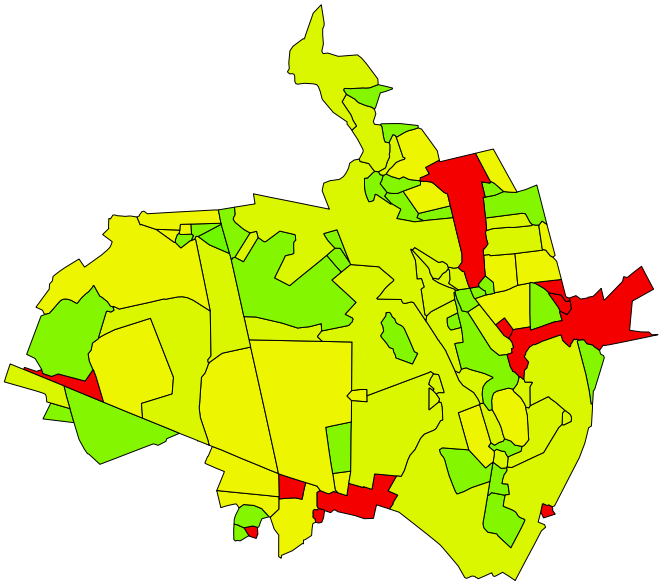


source: 92 patches

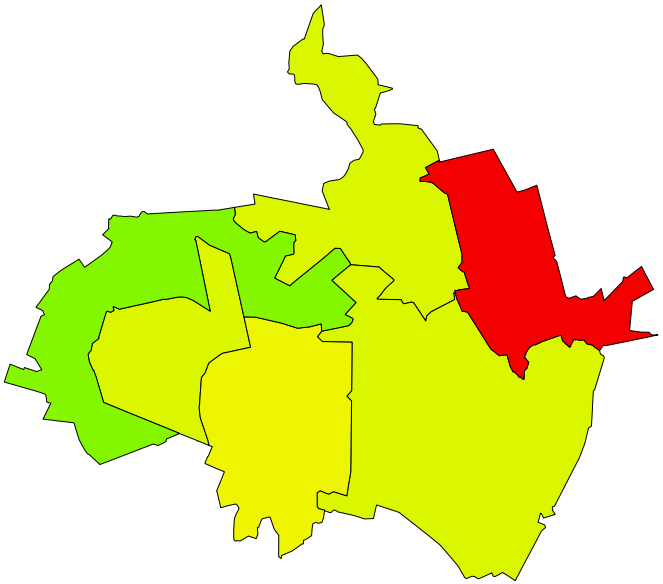


target: 6 patches

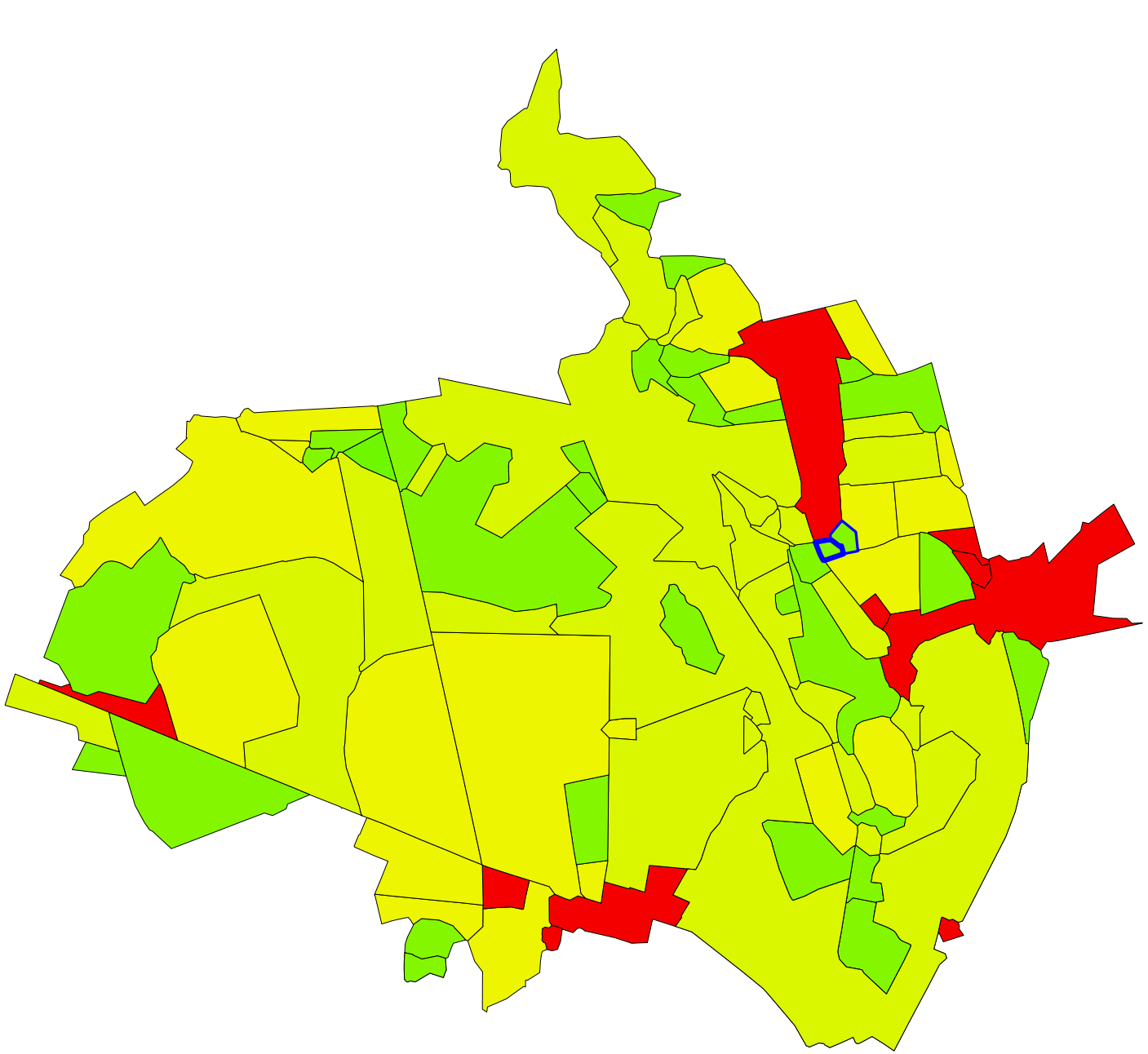


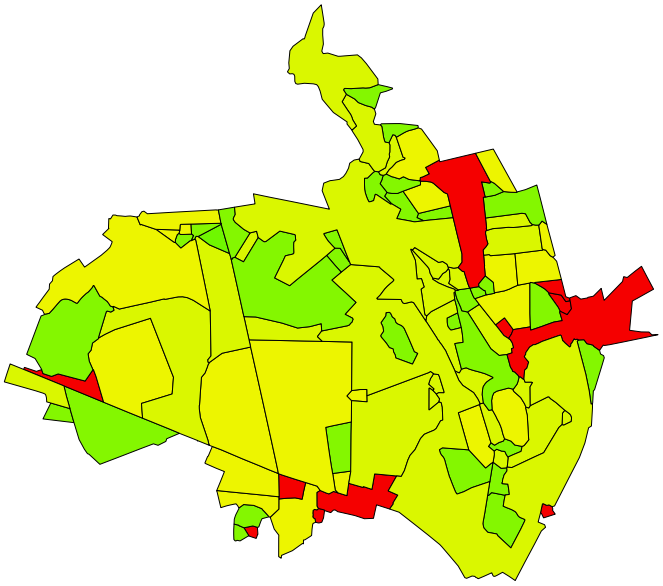


source: 92 patches

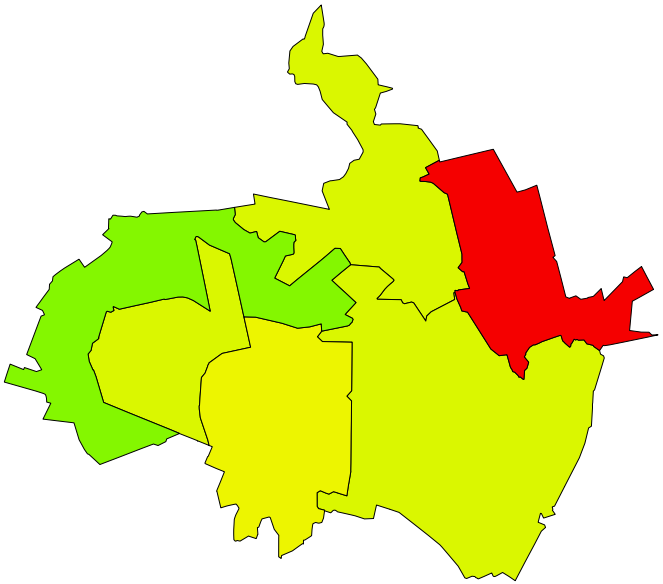


target: 6 patches

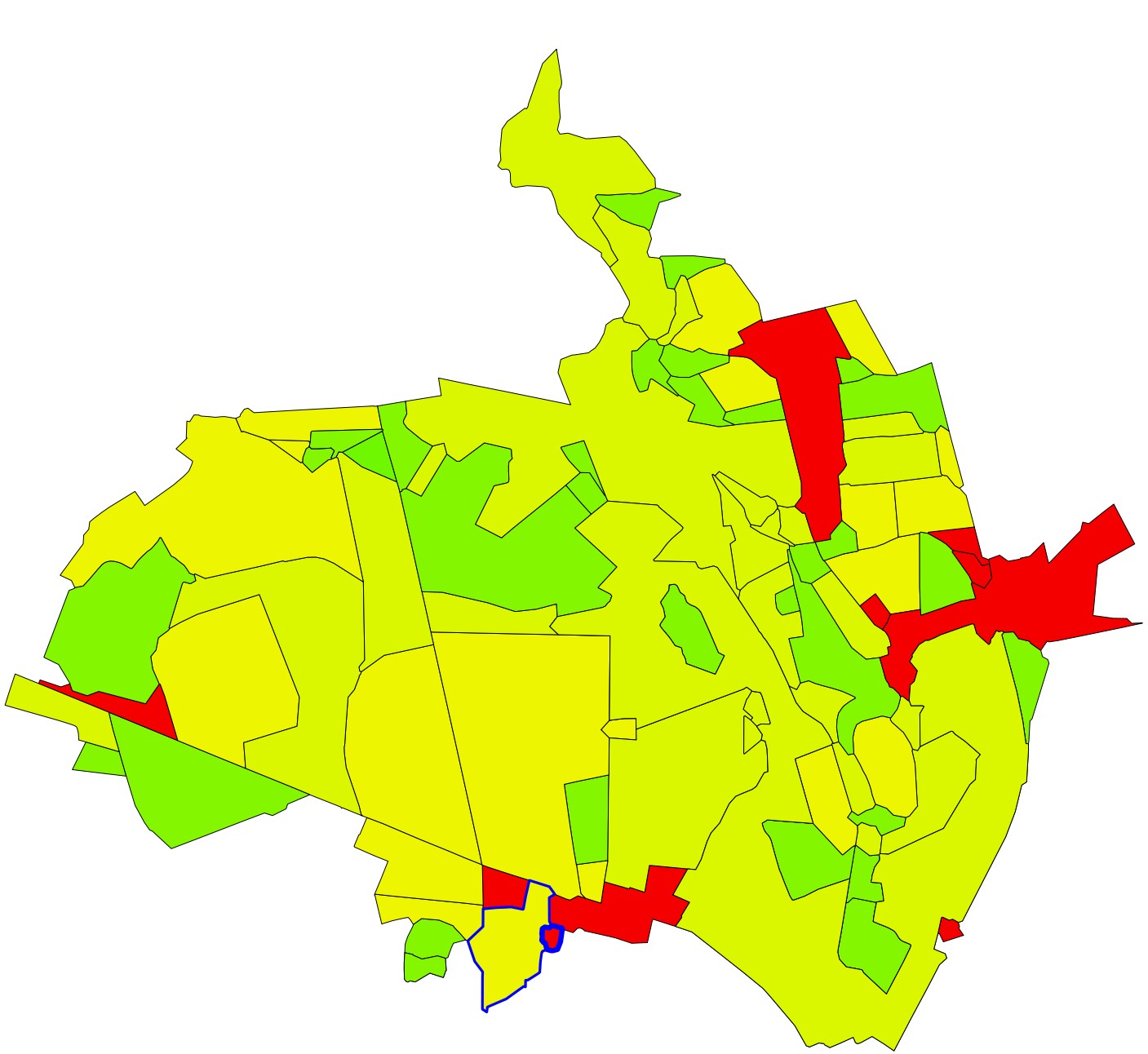


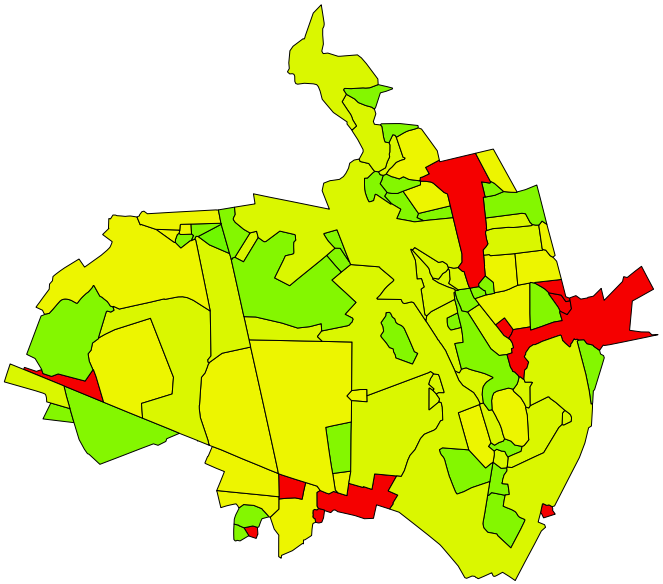


source: 92 patches

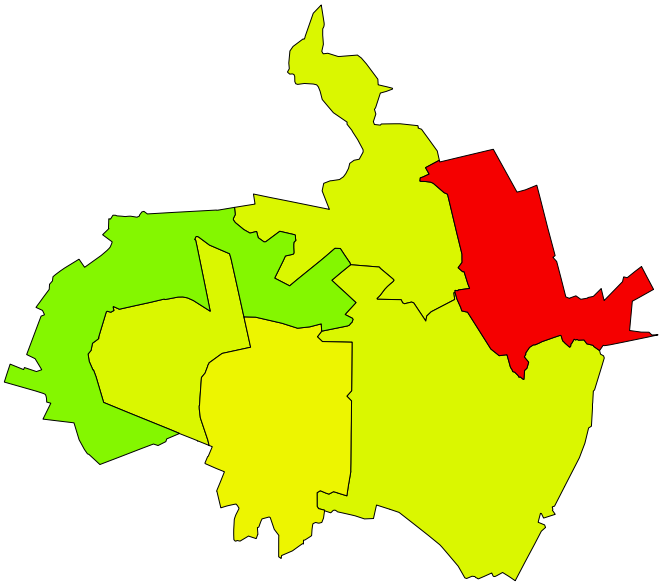


target: 6 patches

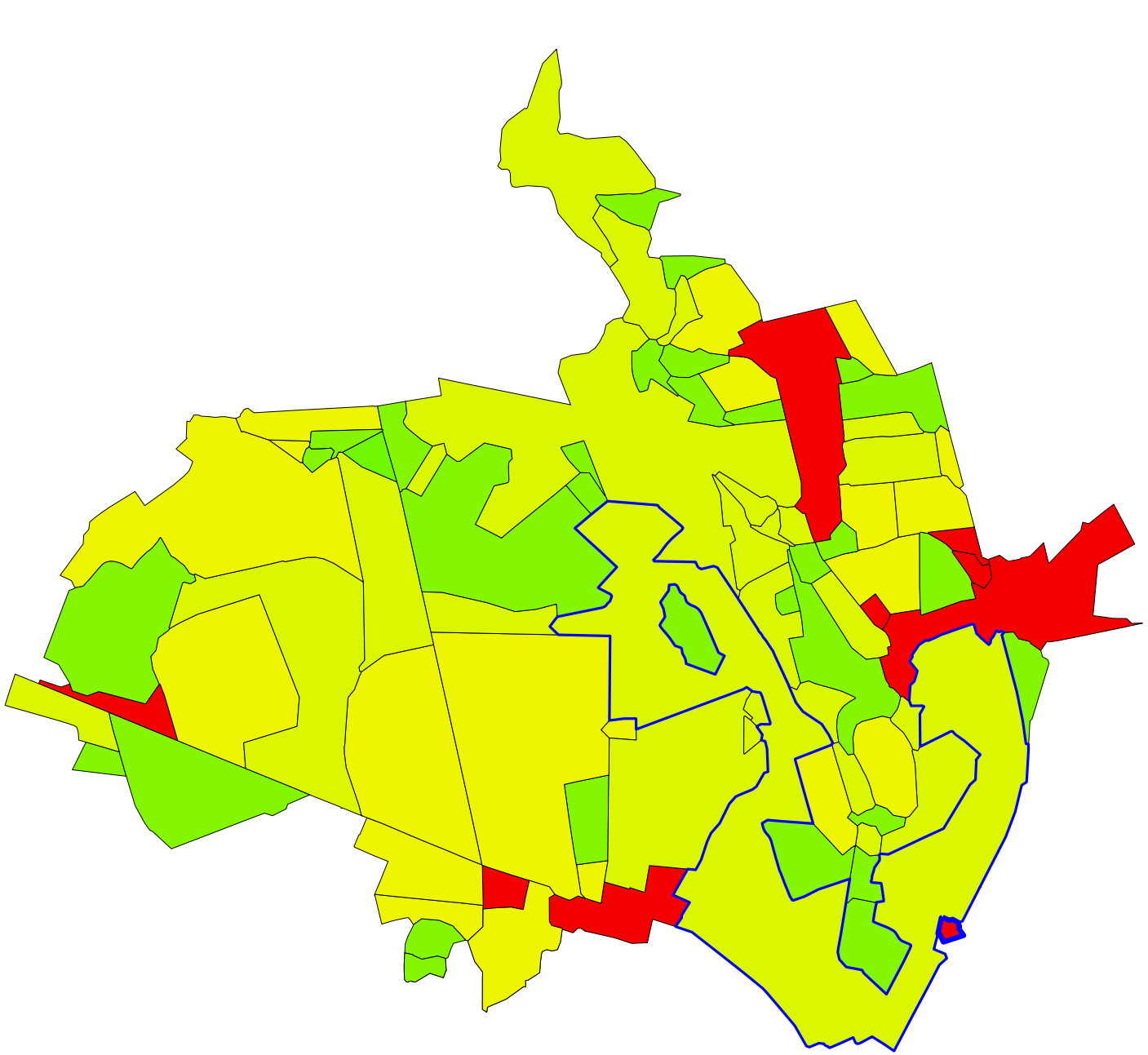


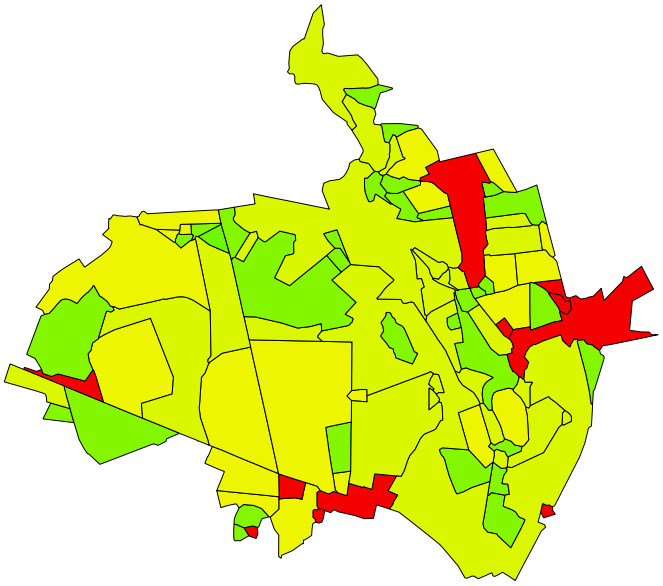


source: 92 patches

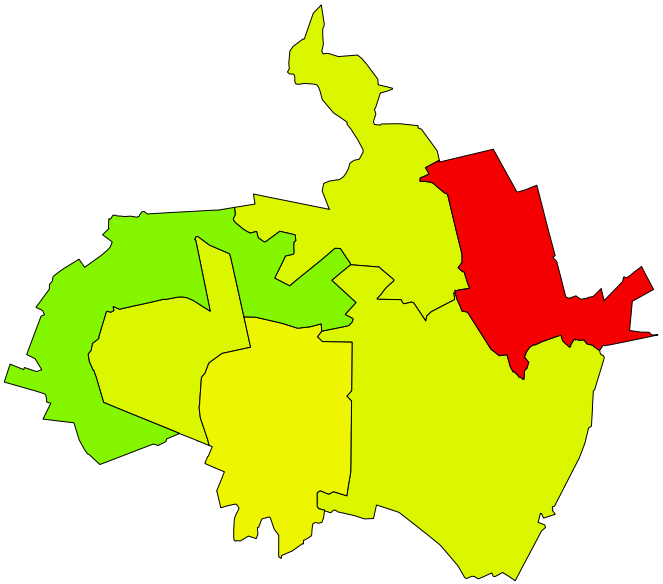


target: 6 patches

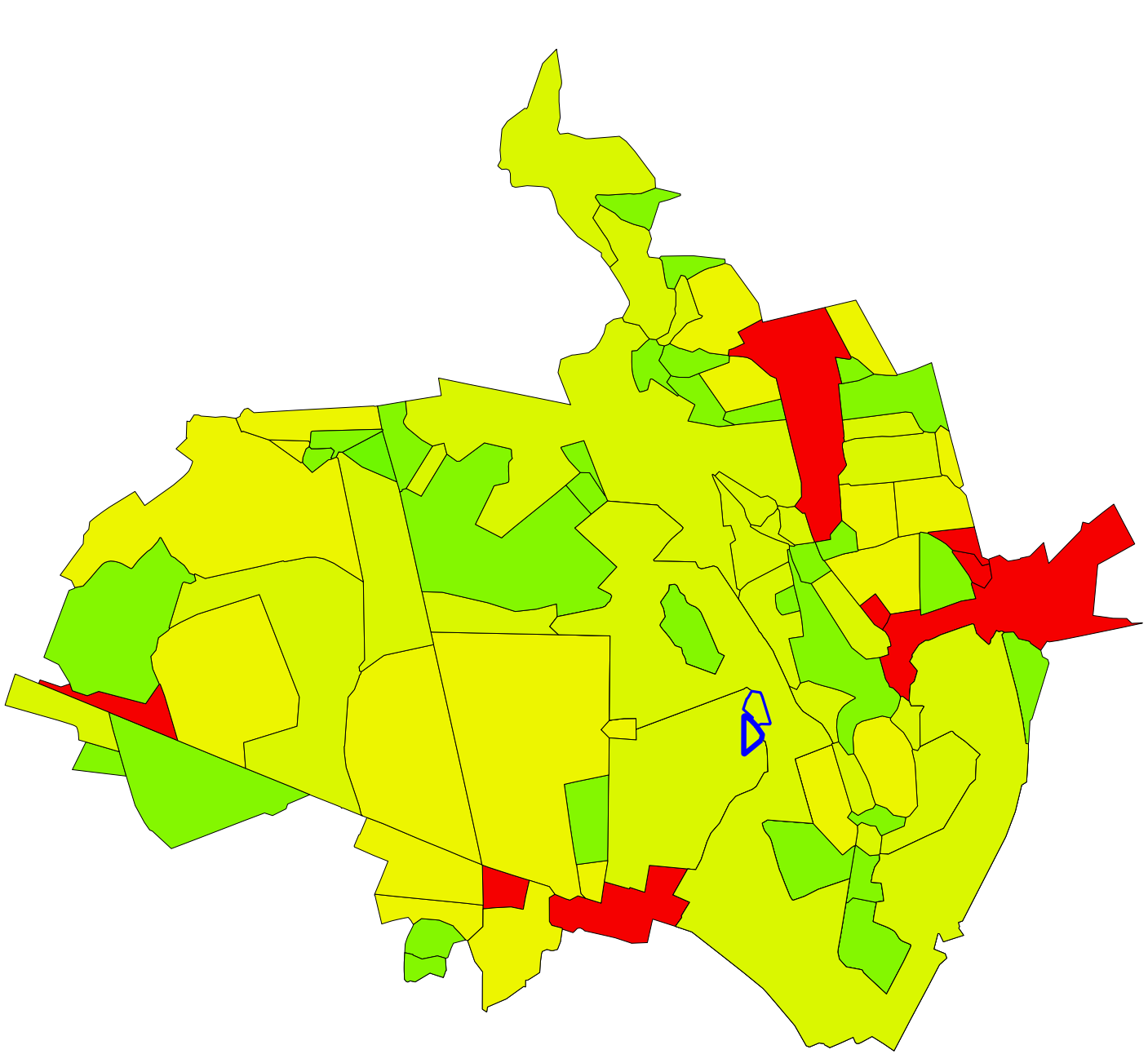


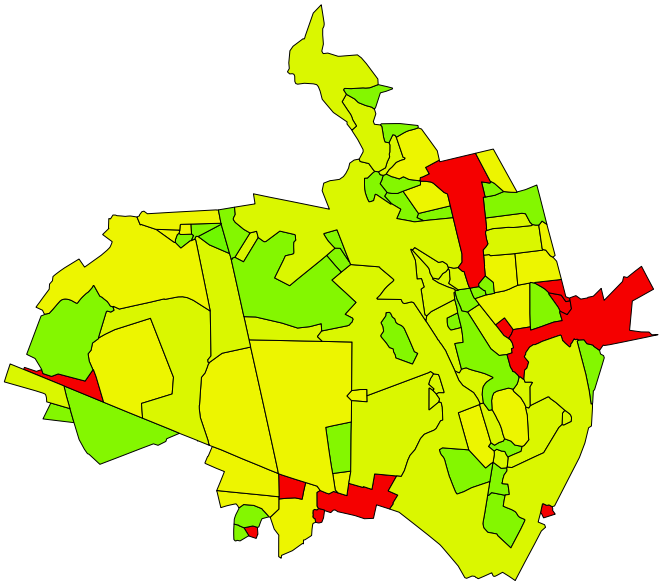


source: 92 patches

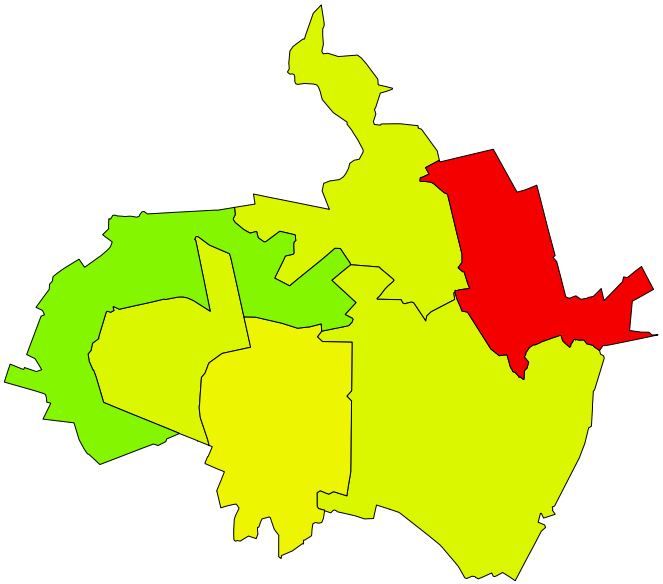


target: 6 patches

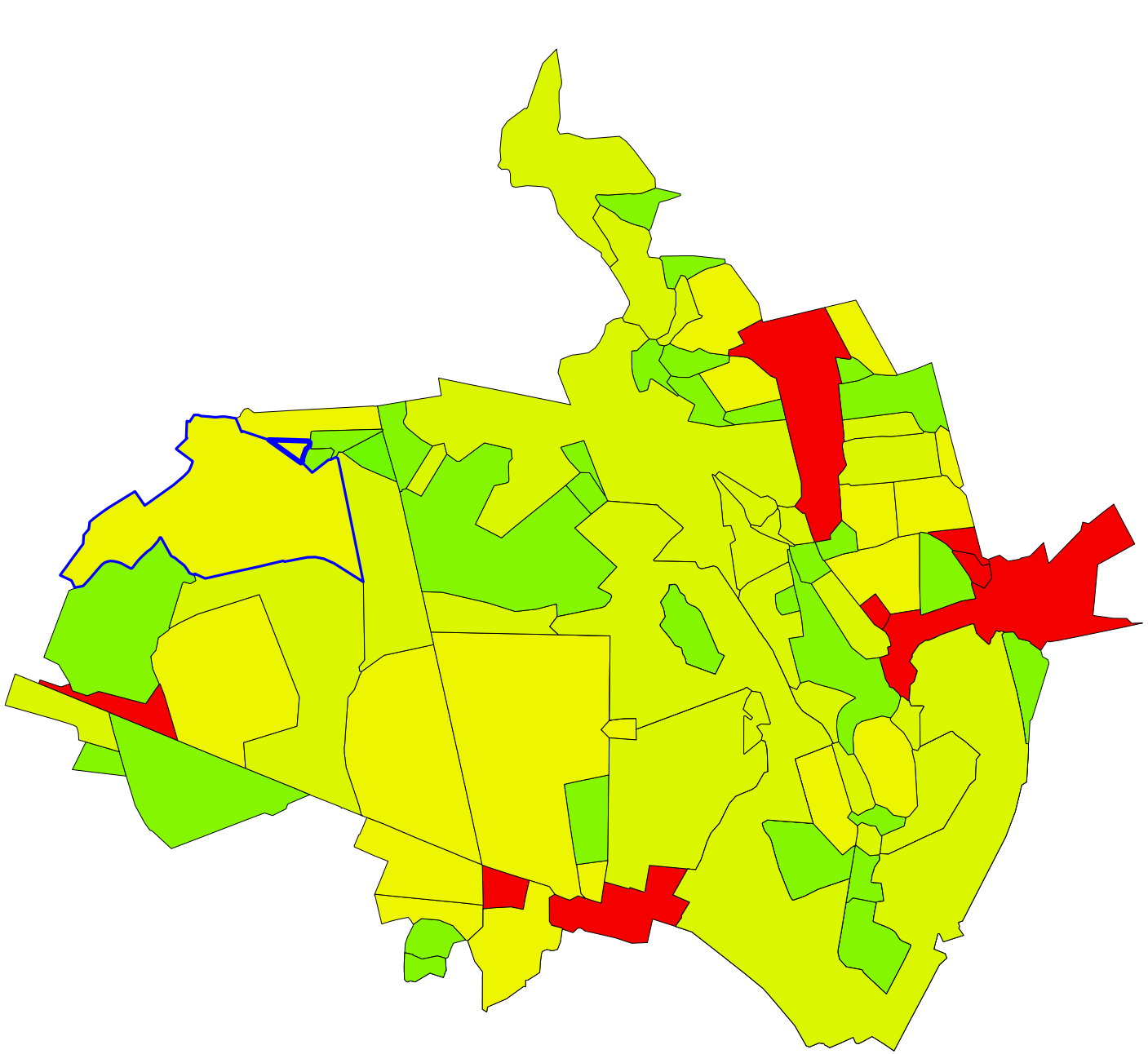


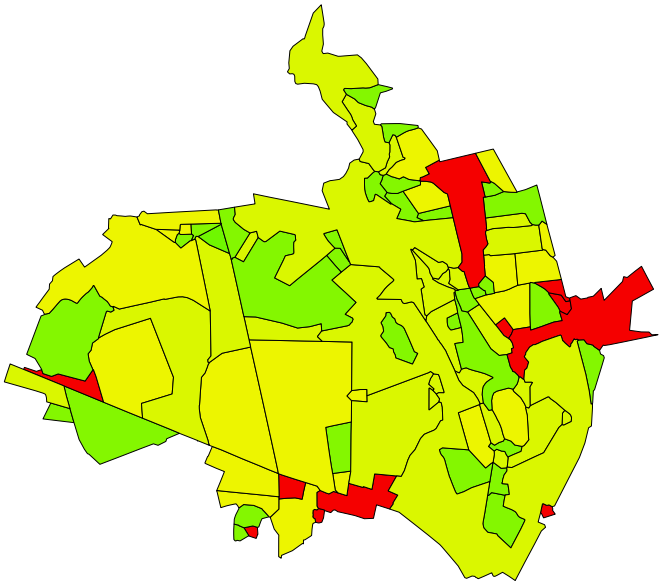


source: 92 patches

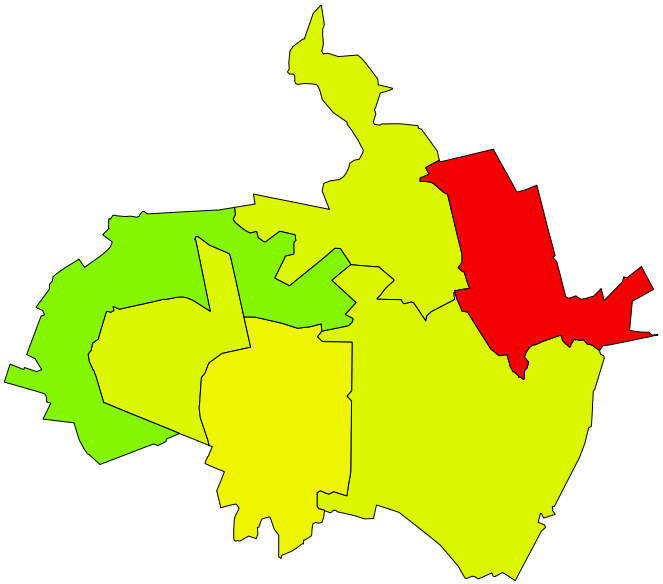


target: 6 patches

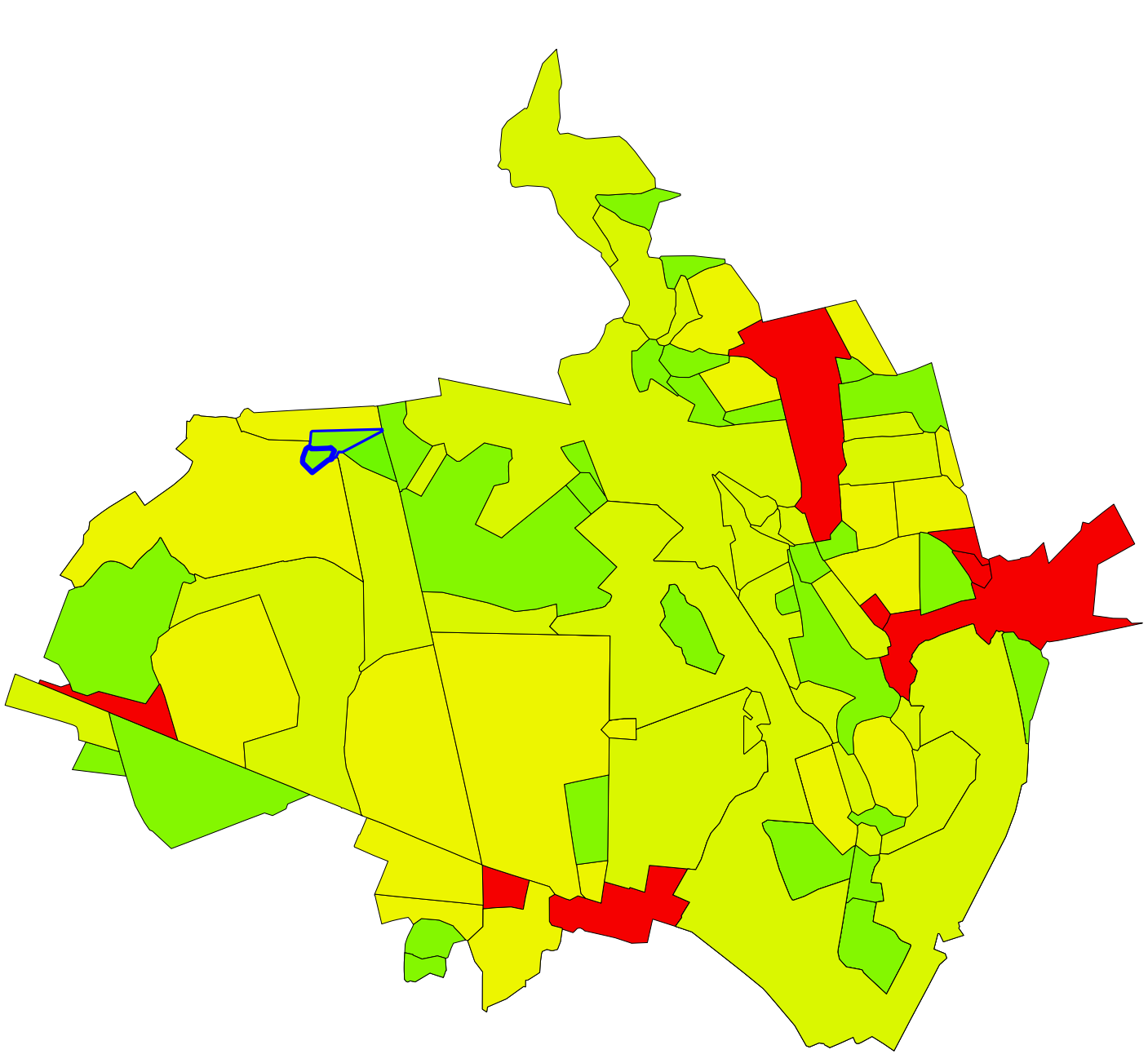


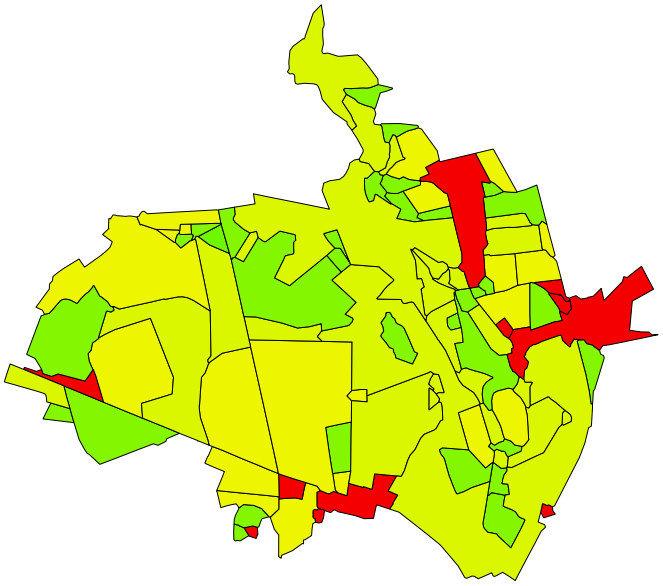


source: 92 patches

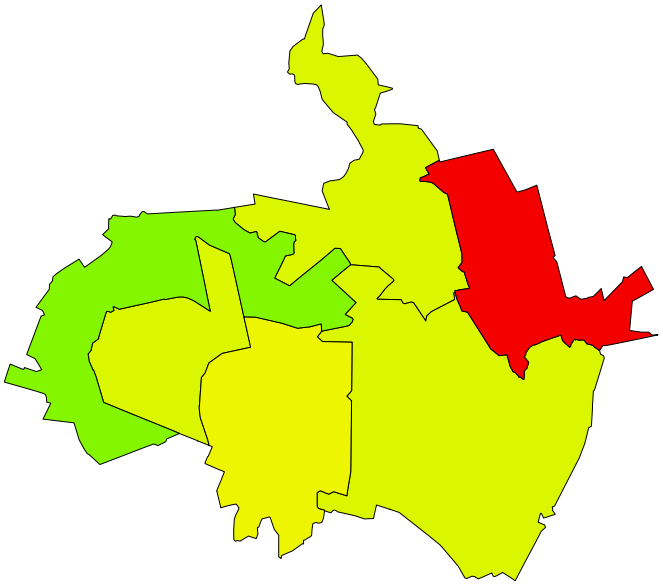


target: 6 patches

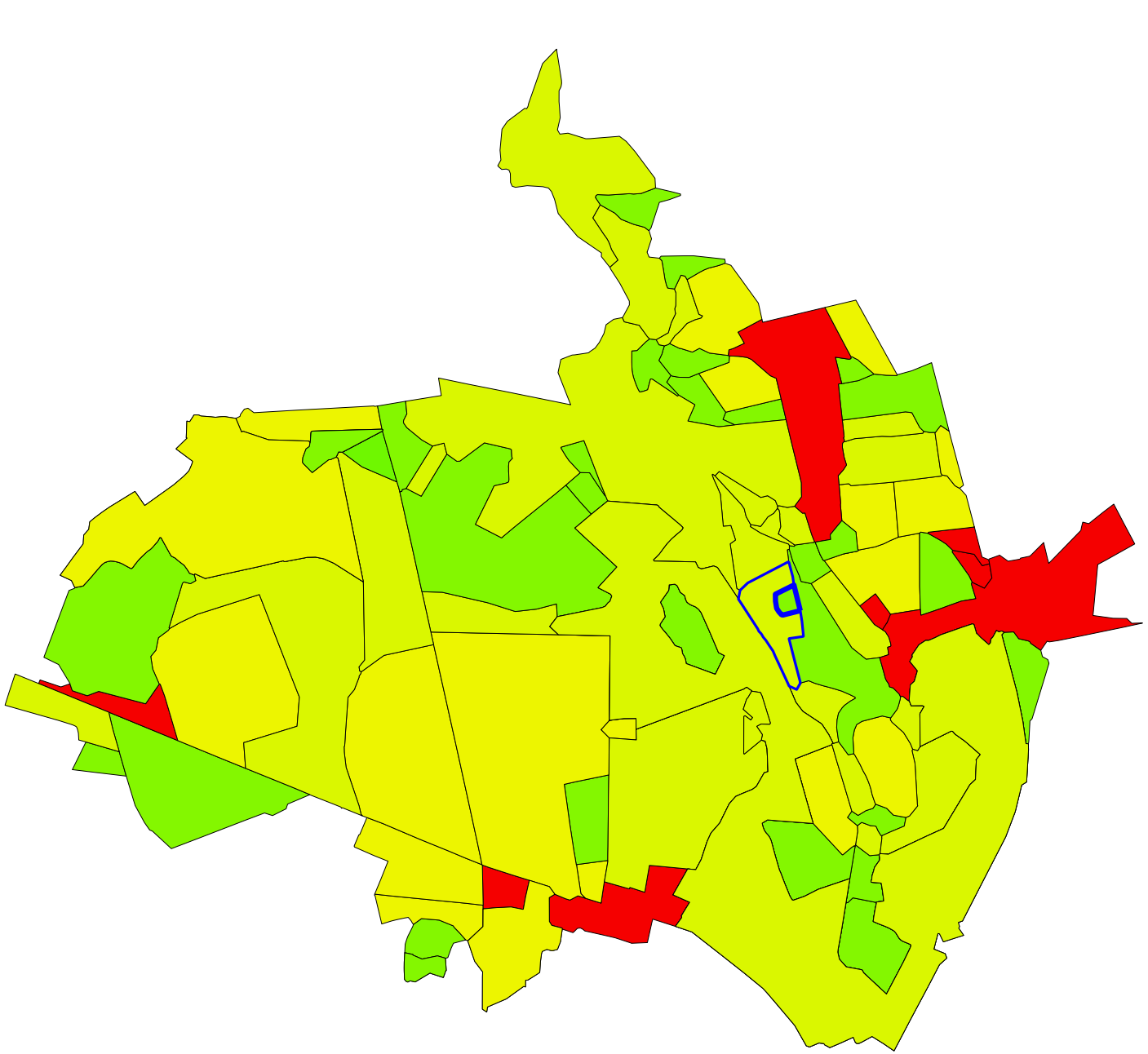


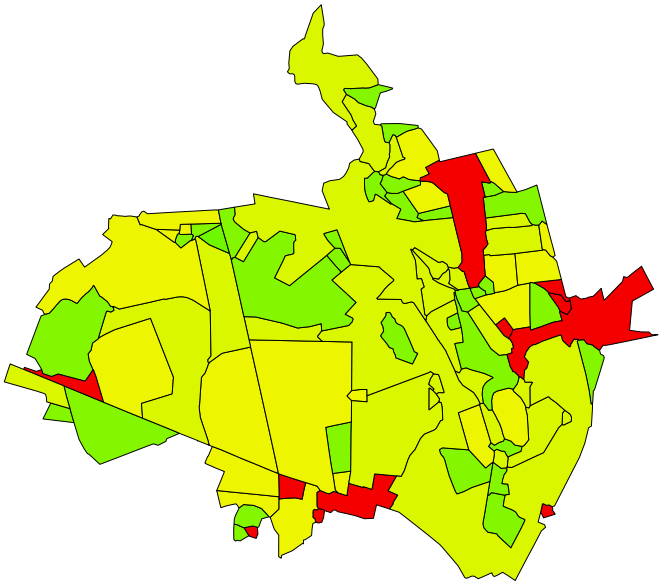


source: 92 patches

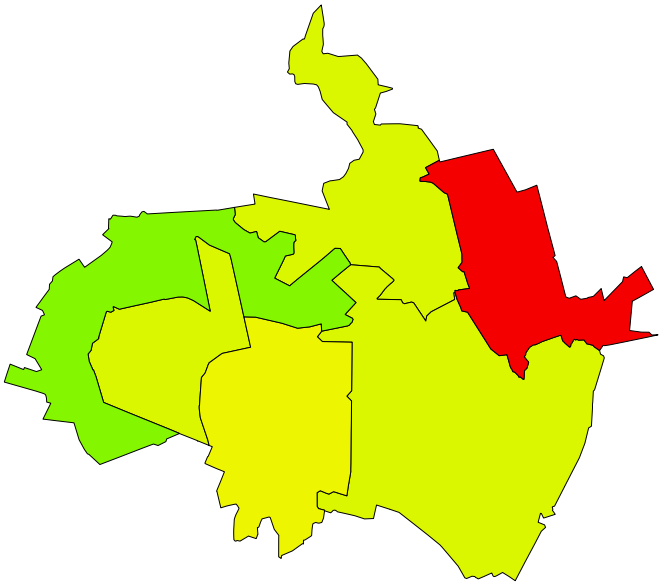


target: 6 patches

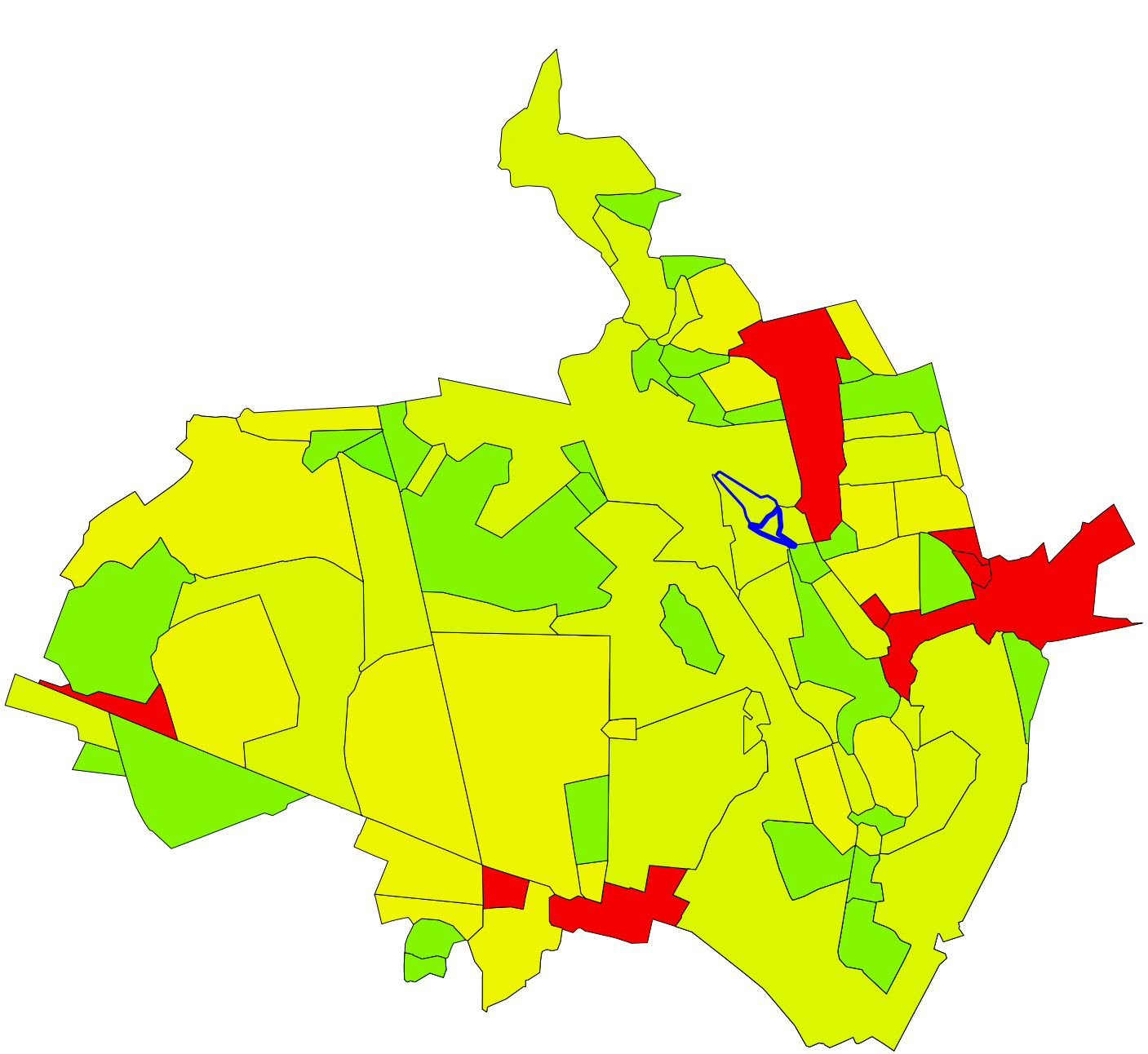


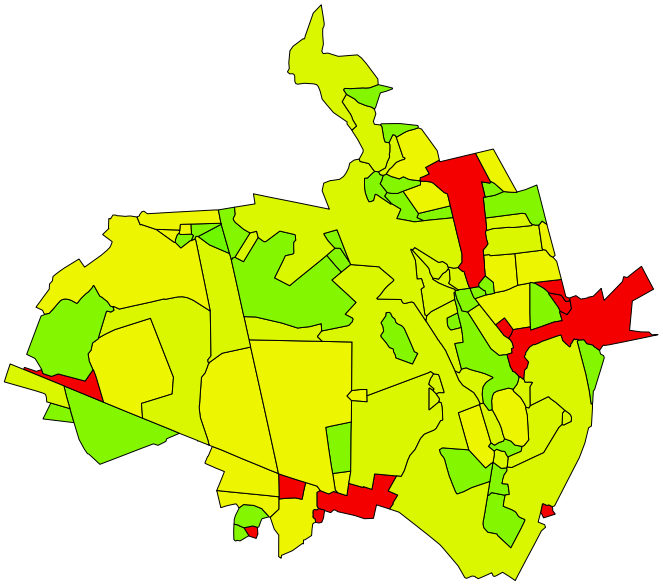


source: 92 patches

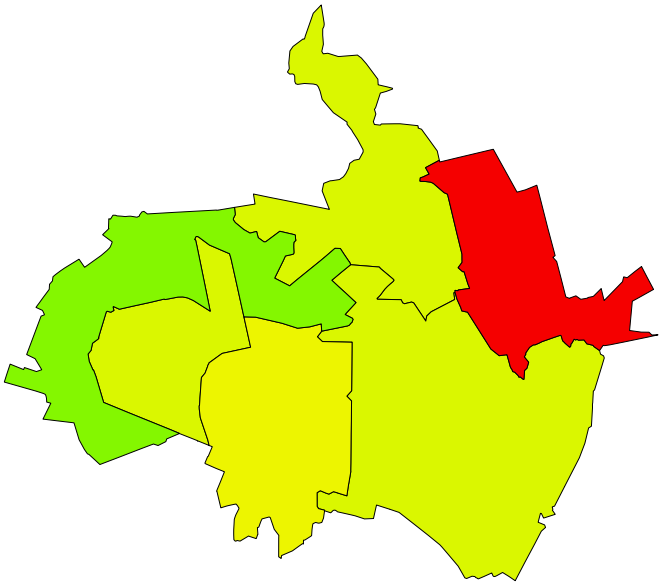


target: 6 patches

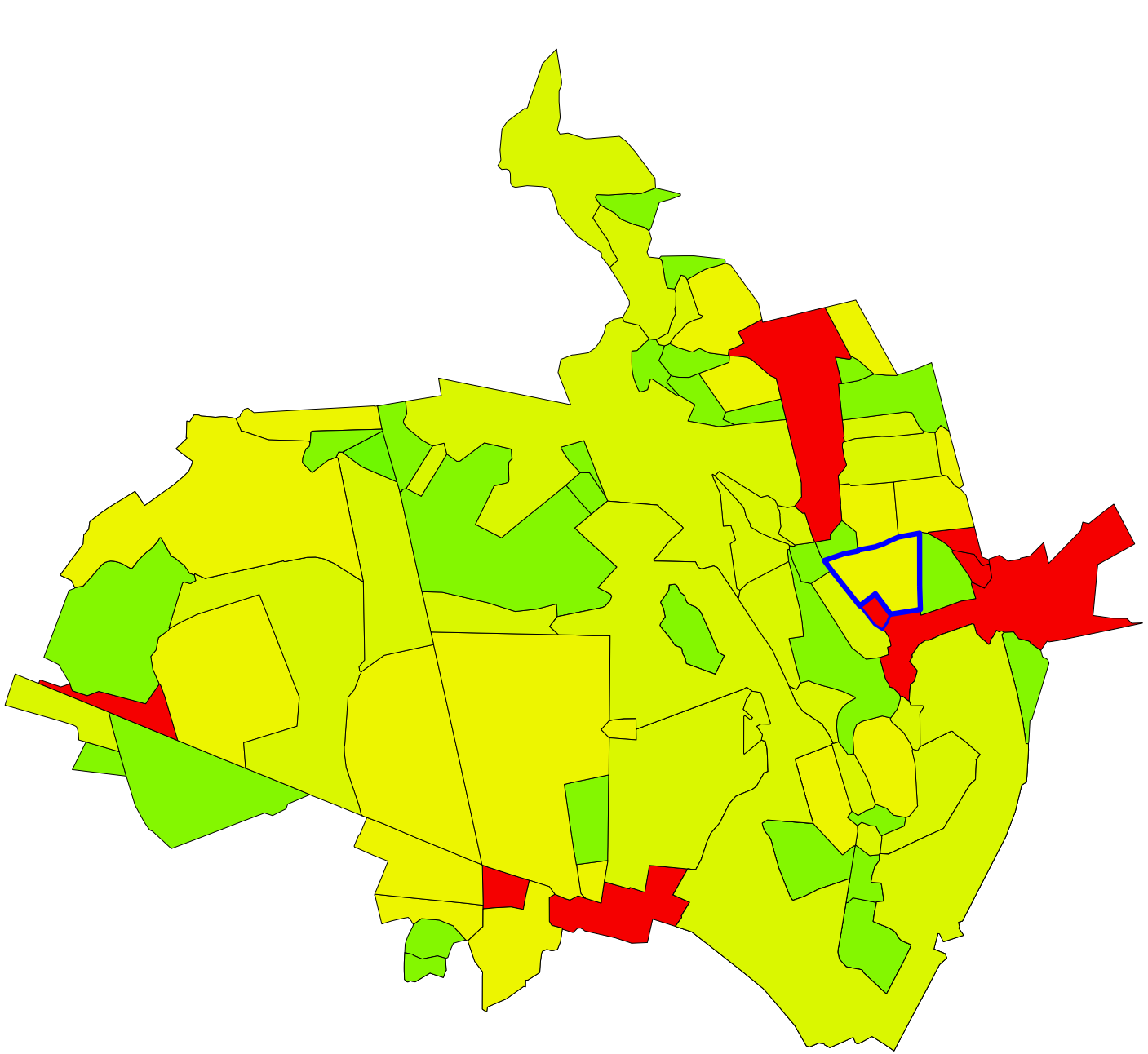


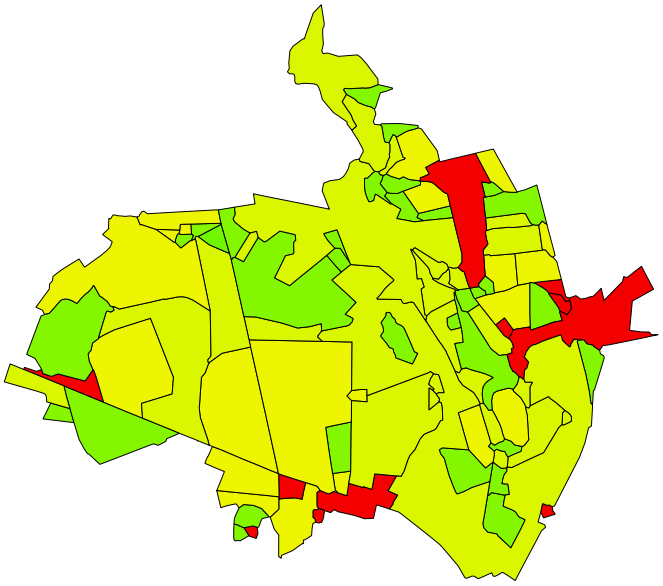


source: 92 patches

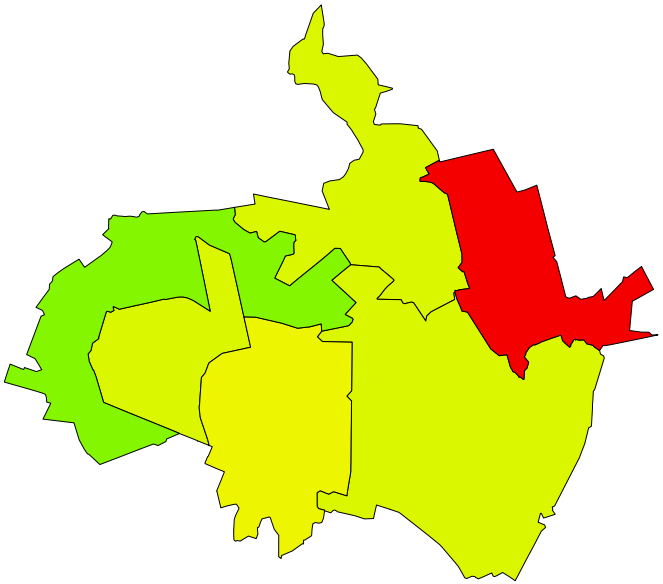


target: 6 patches

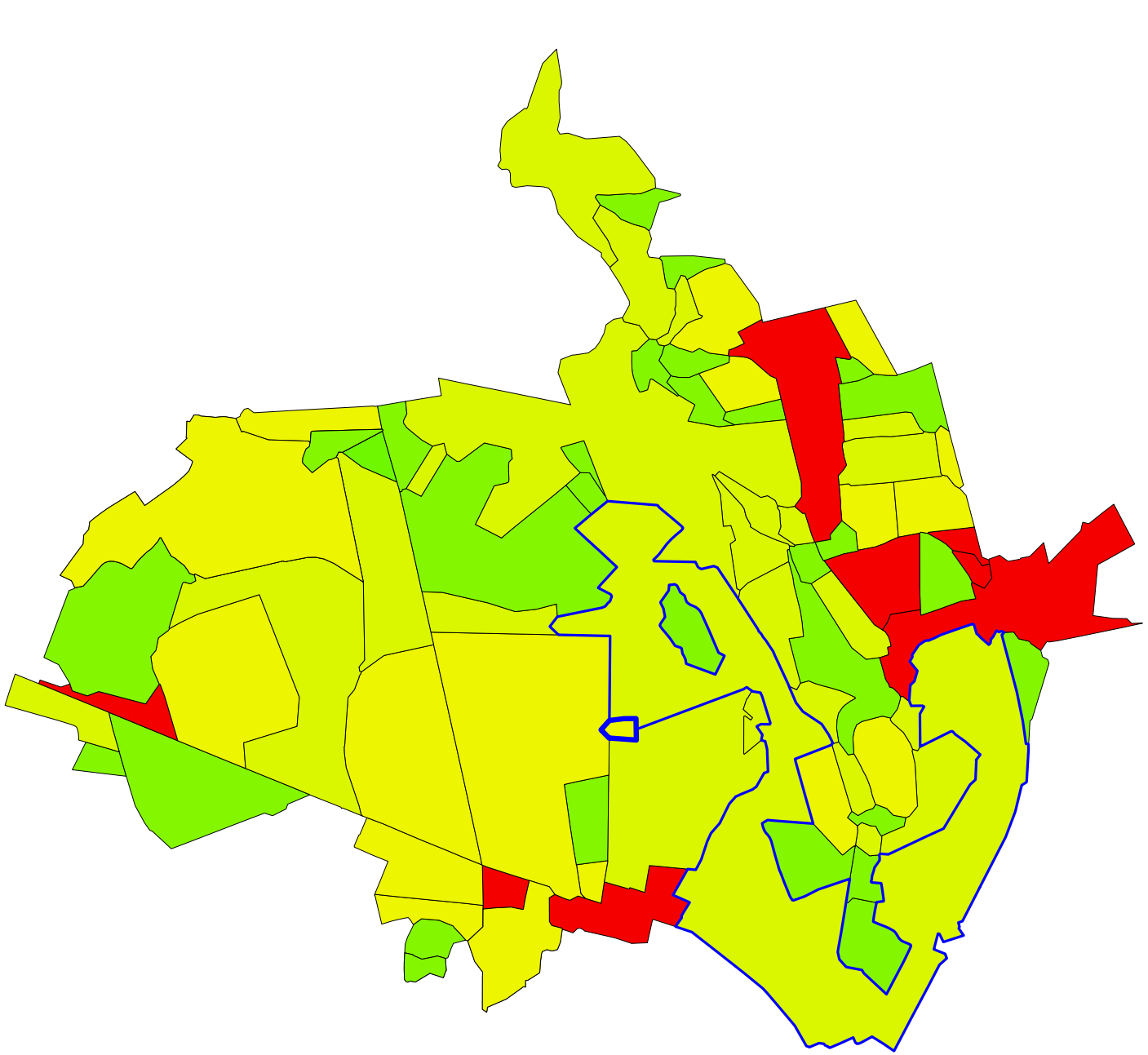


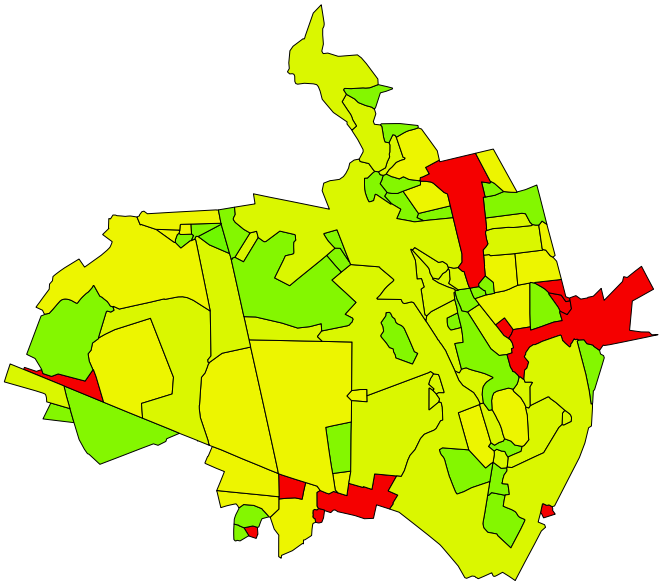


source: 92 patches

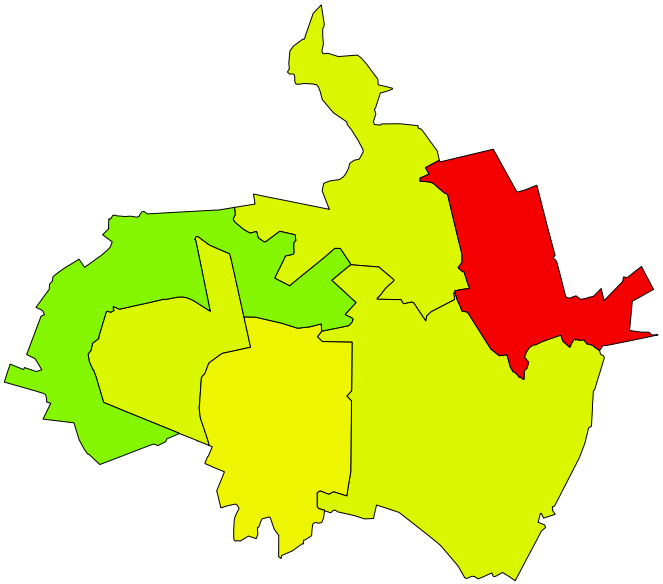


target: 6 patches

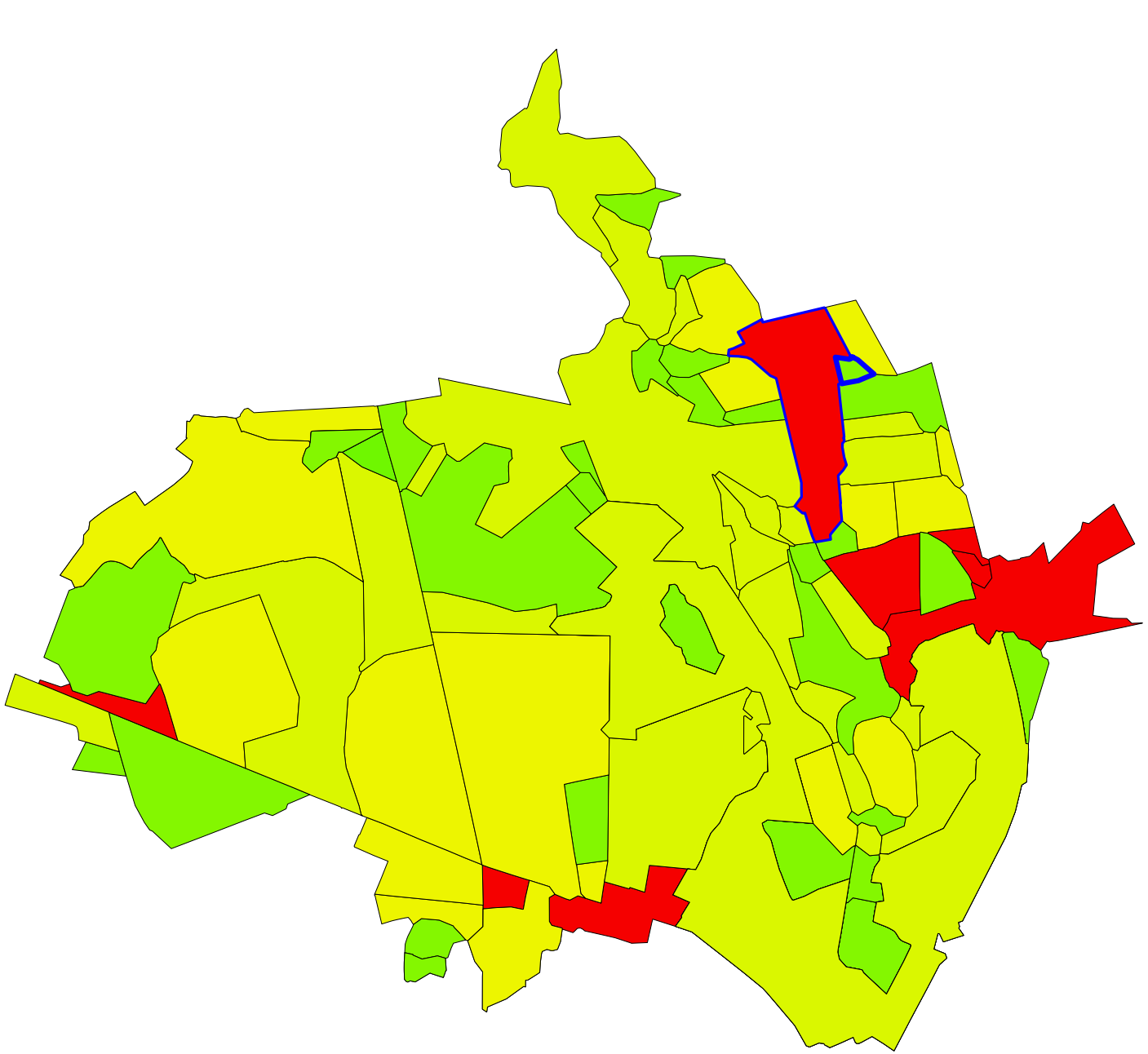


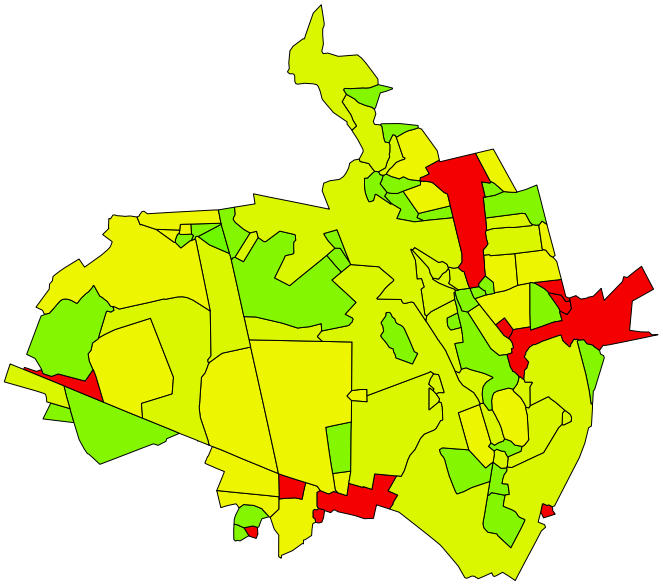


source: 92 patches

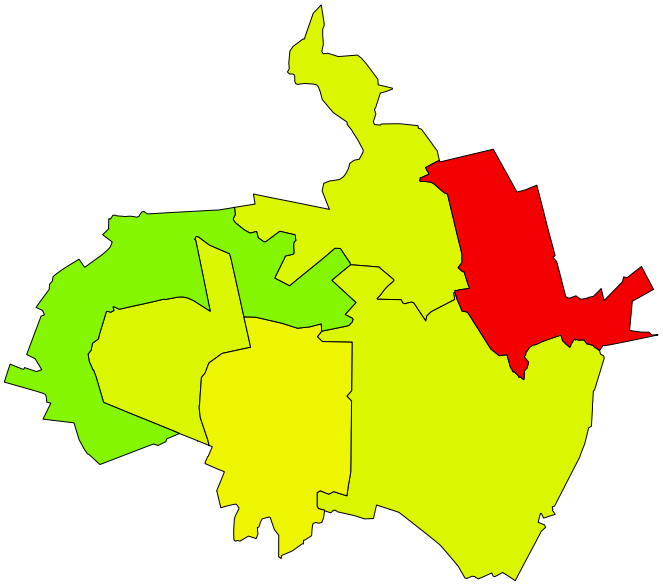


target: 6 patches

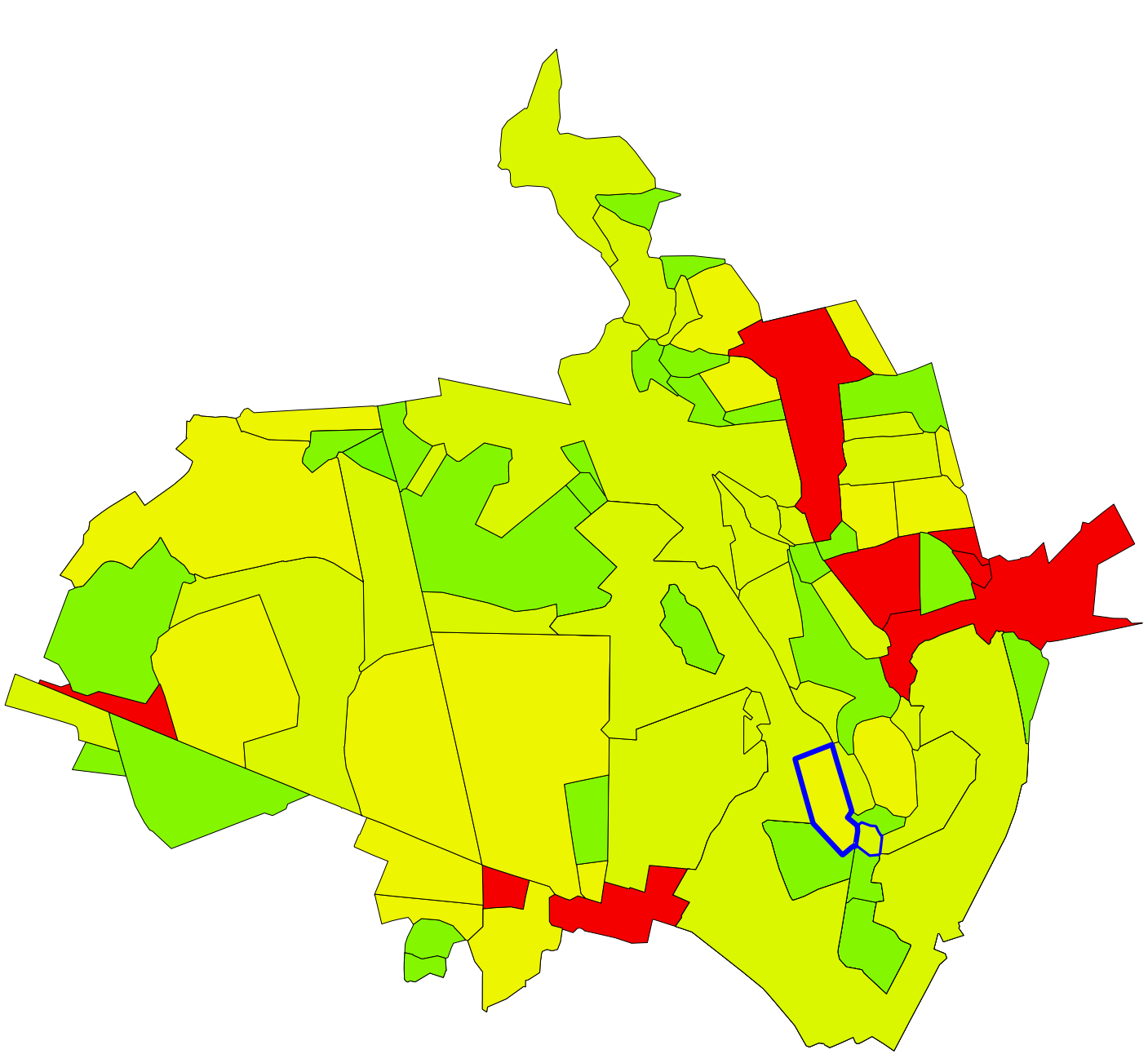


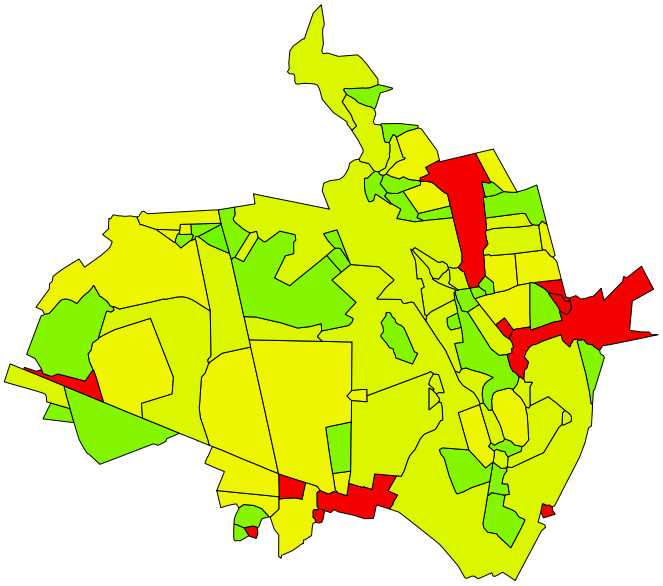


source: 92 patches

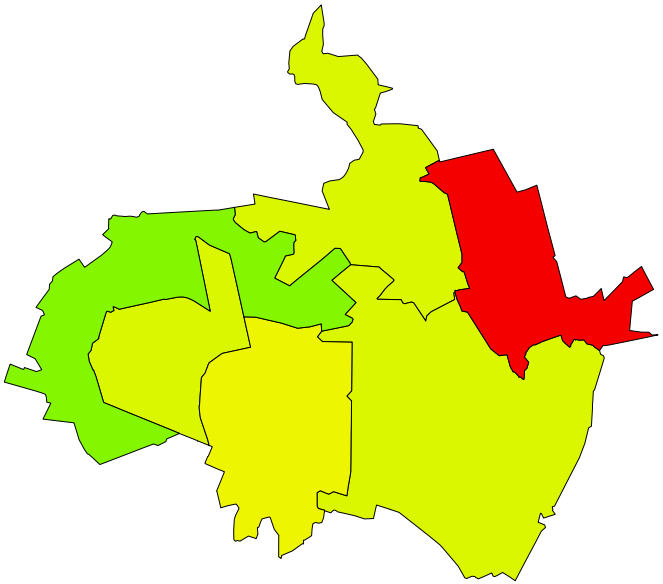


target: 6 patches

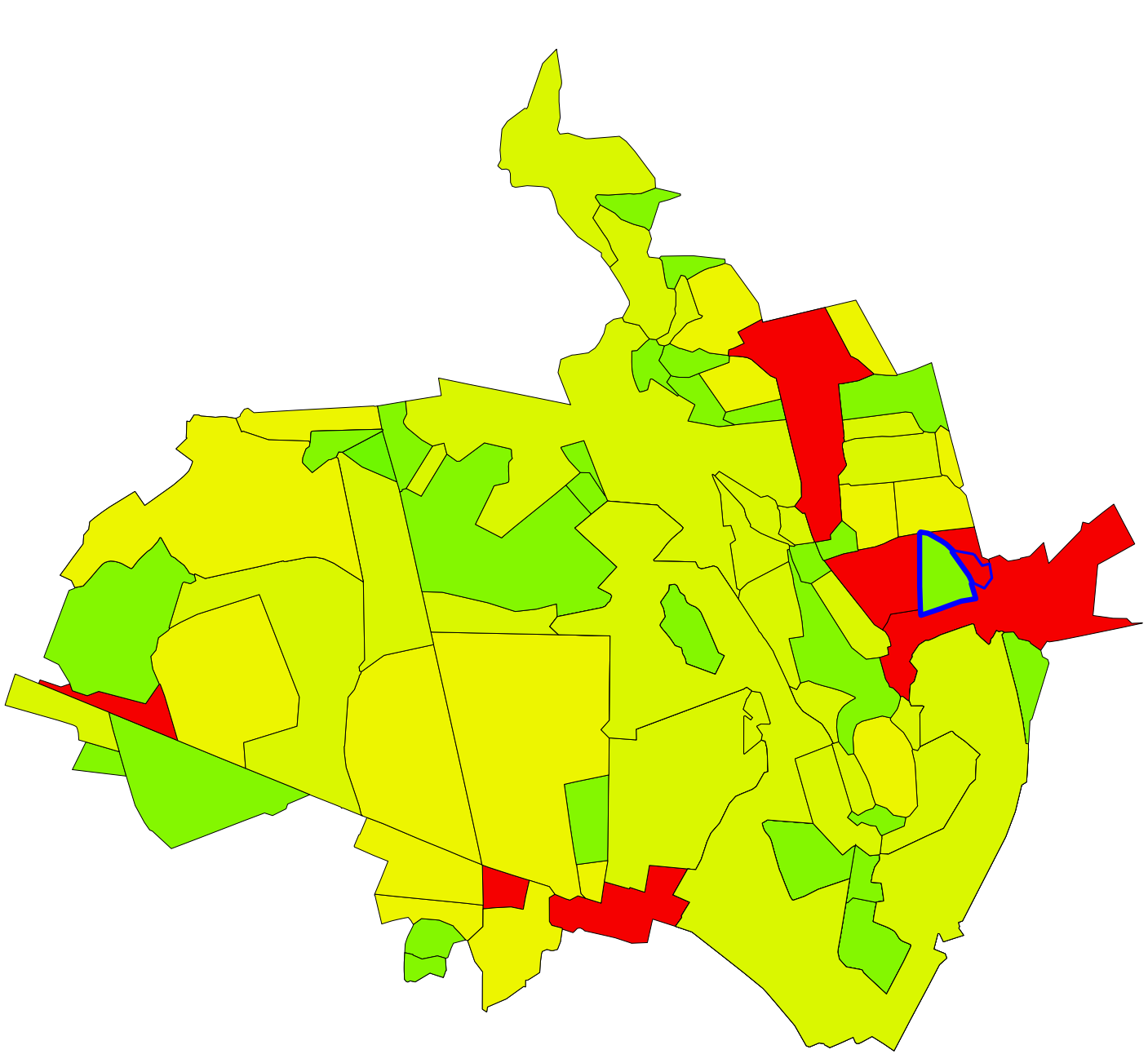


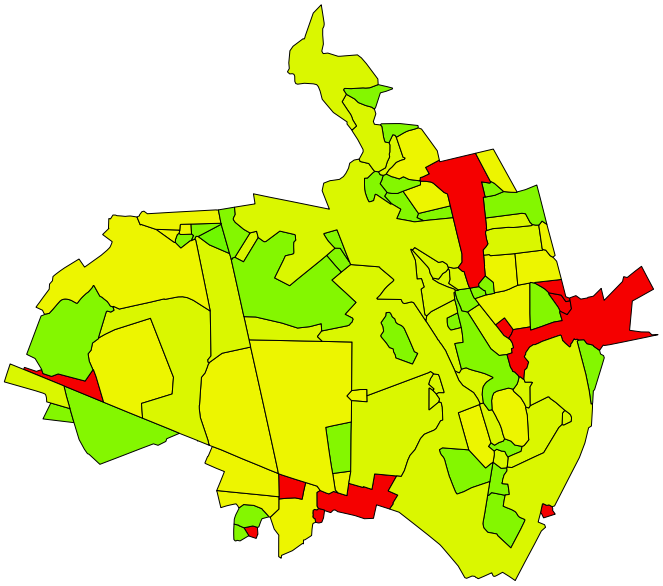


source: 92 patches

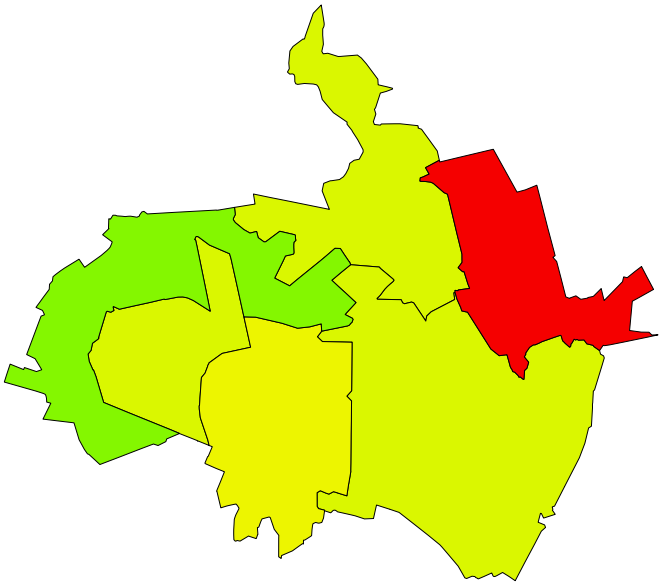


target: 6 patches

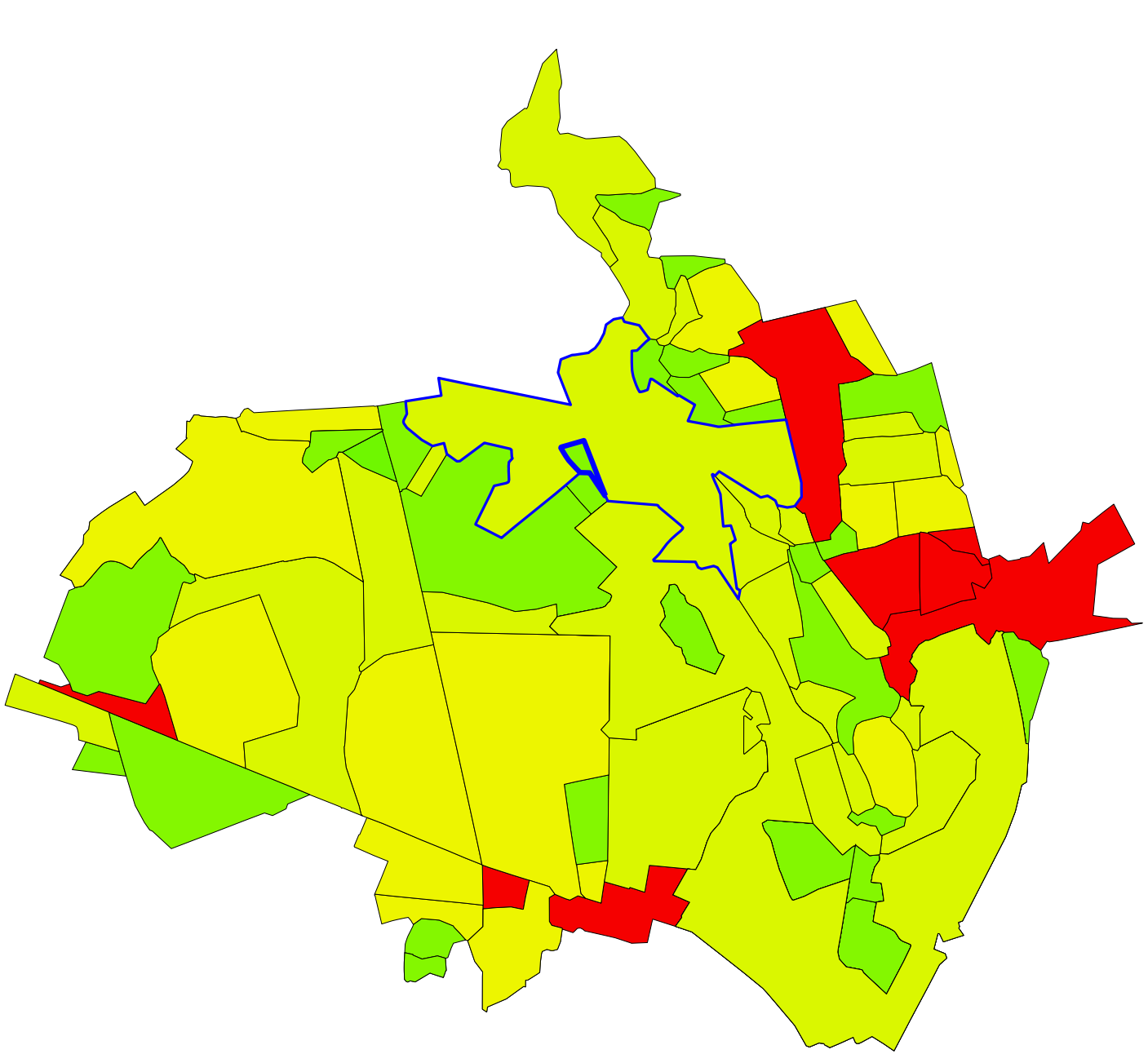


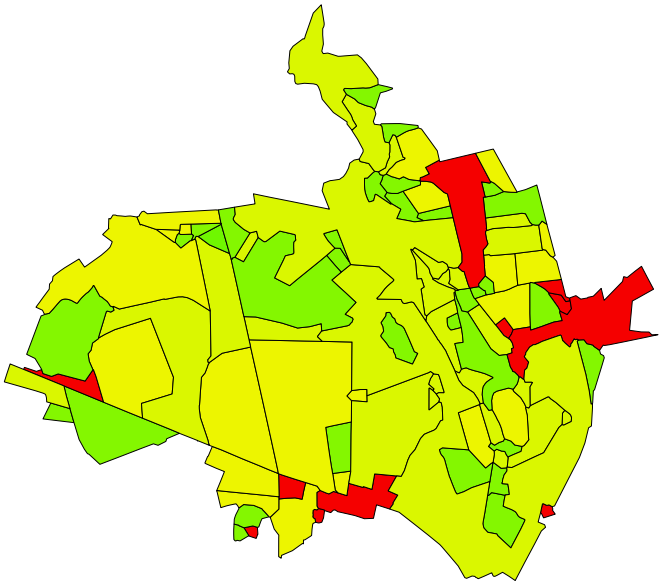


source: 92 patches

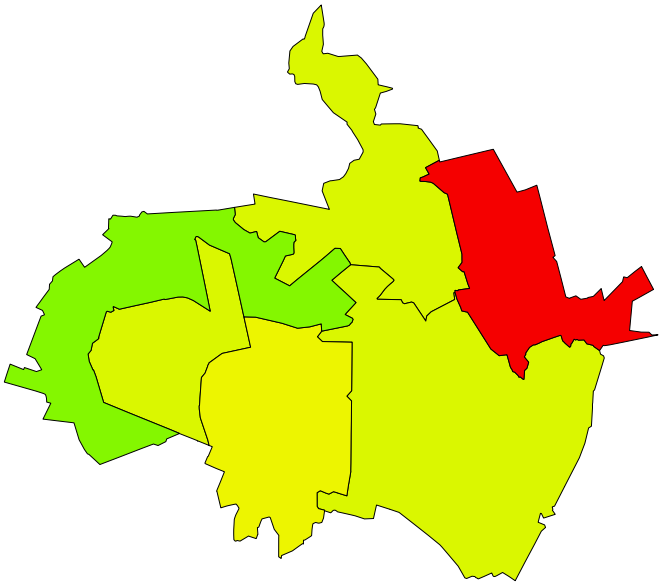


target: 6 patches

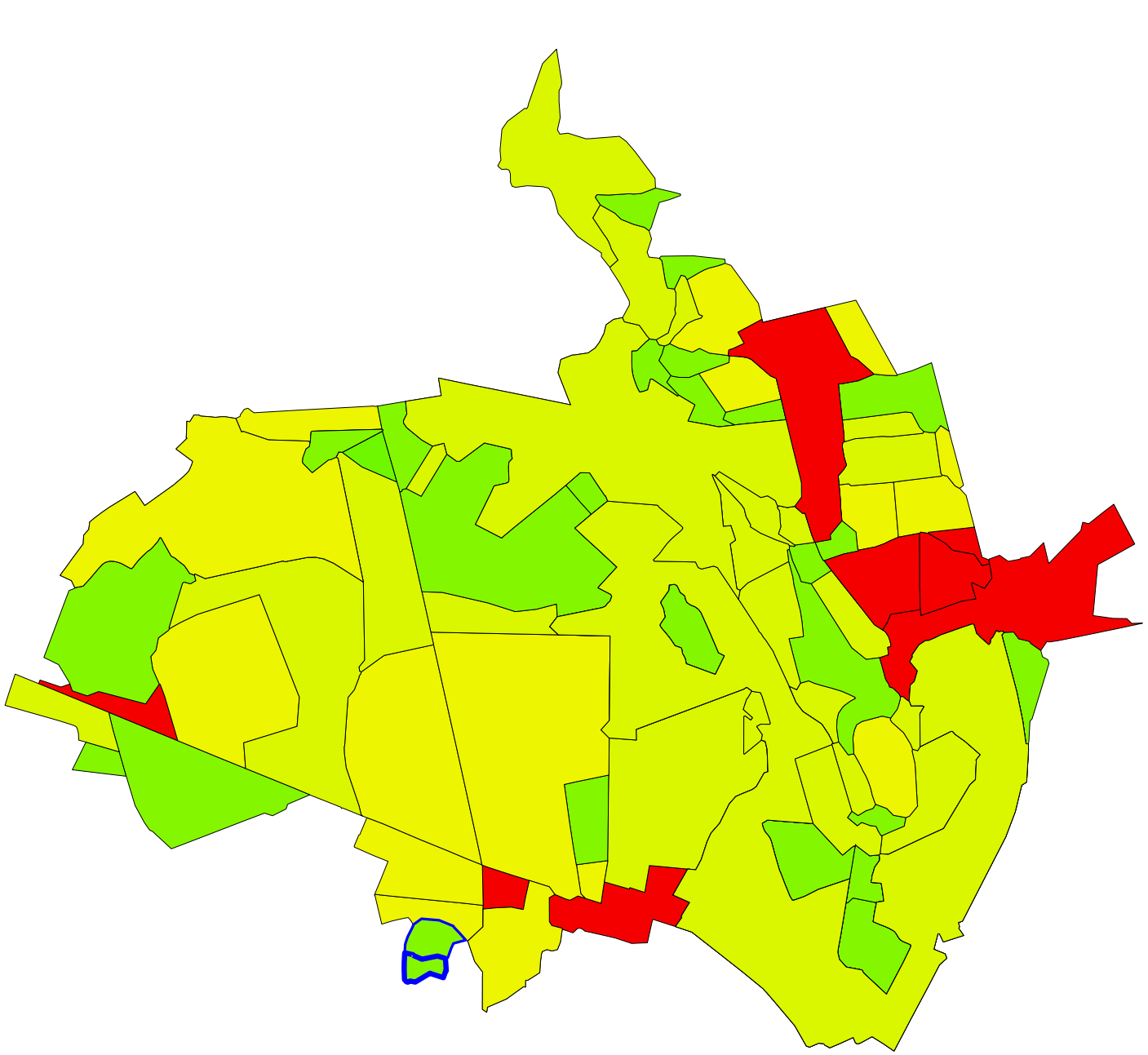


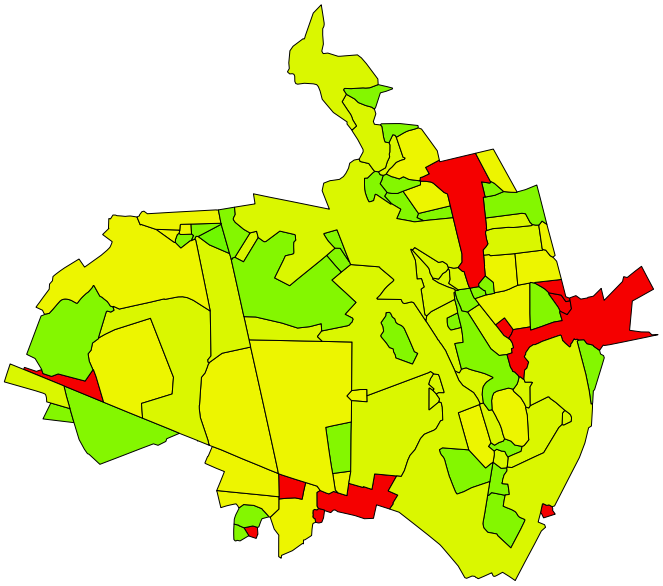


source: 92 patches

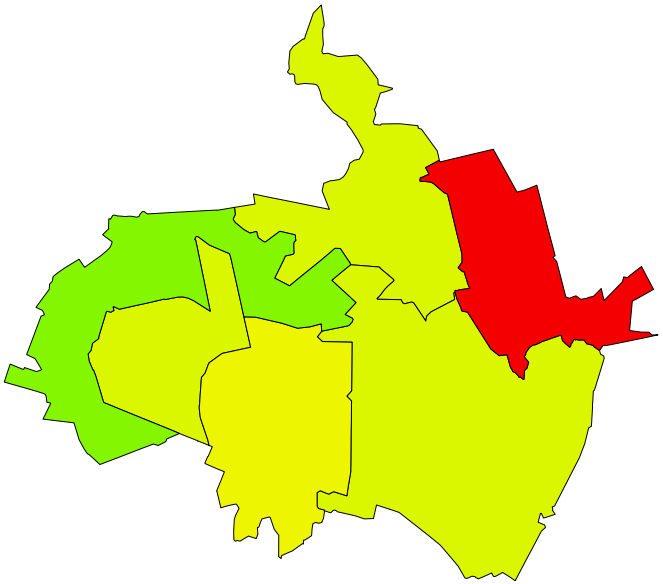


target: 6 patches

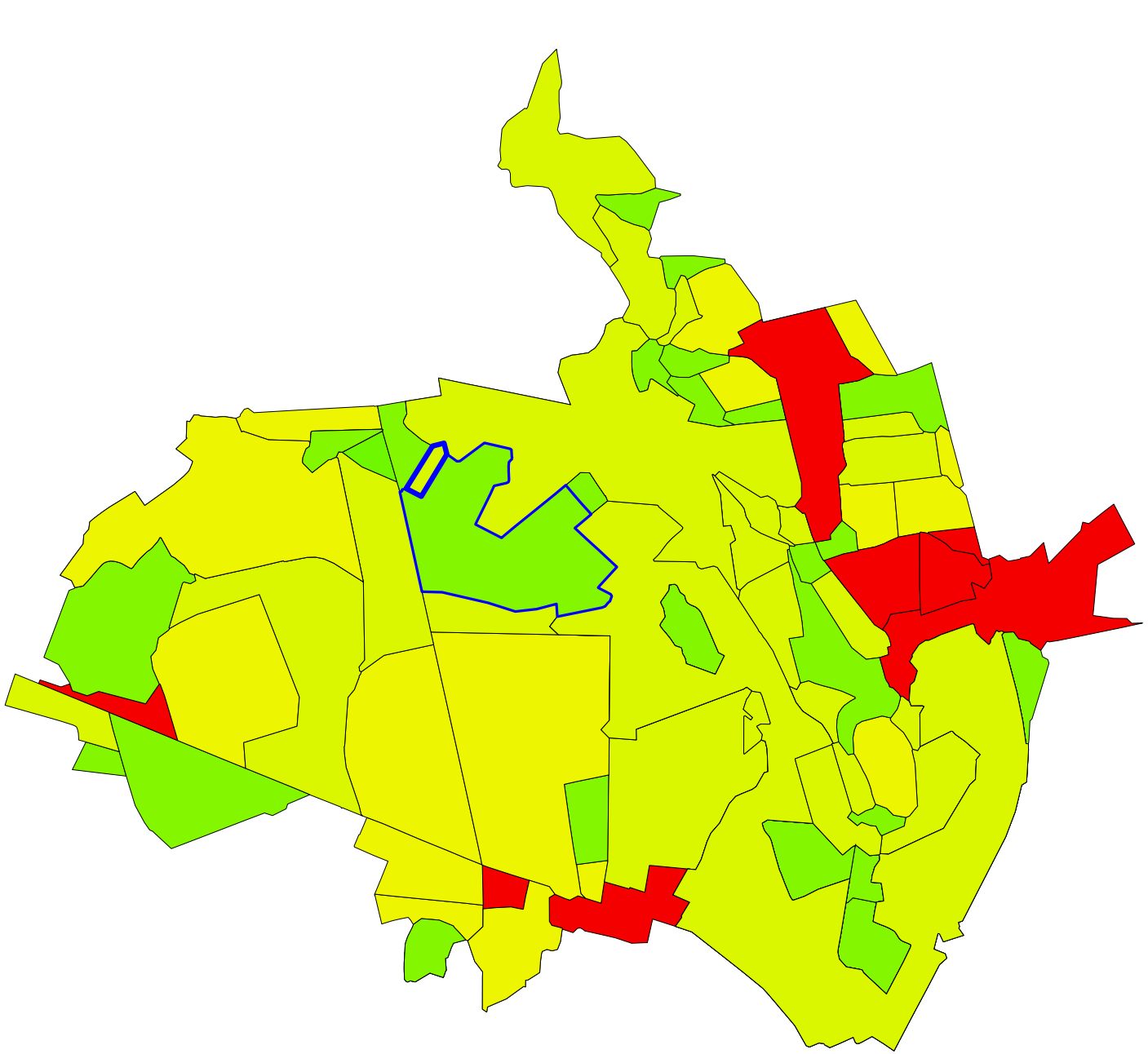


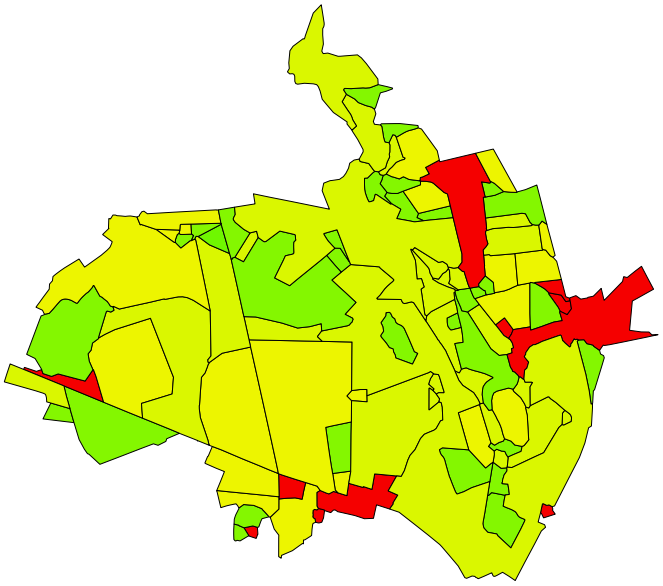


source: 92 patches

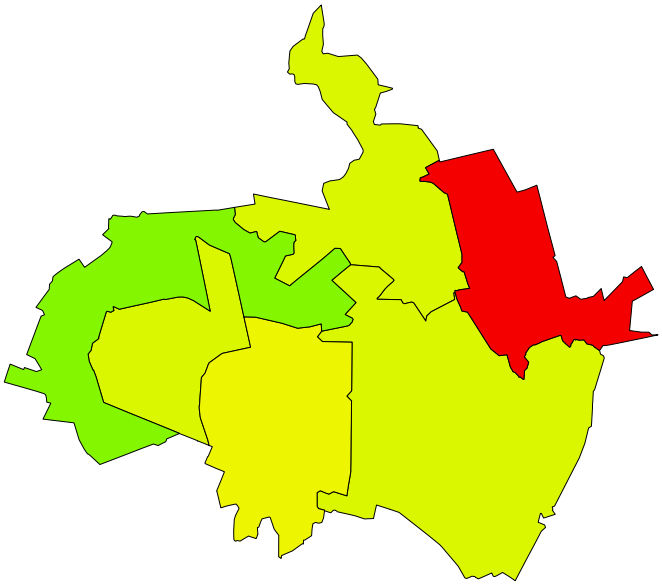


target: 6 patches

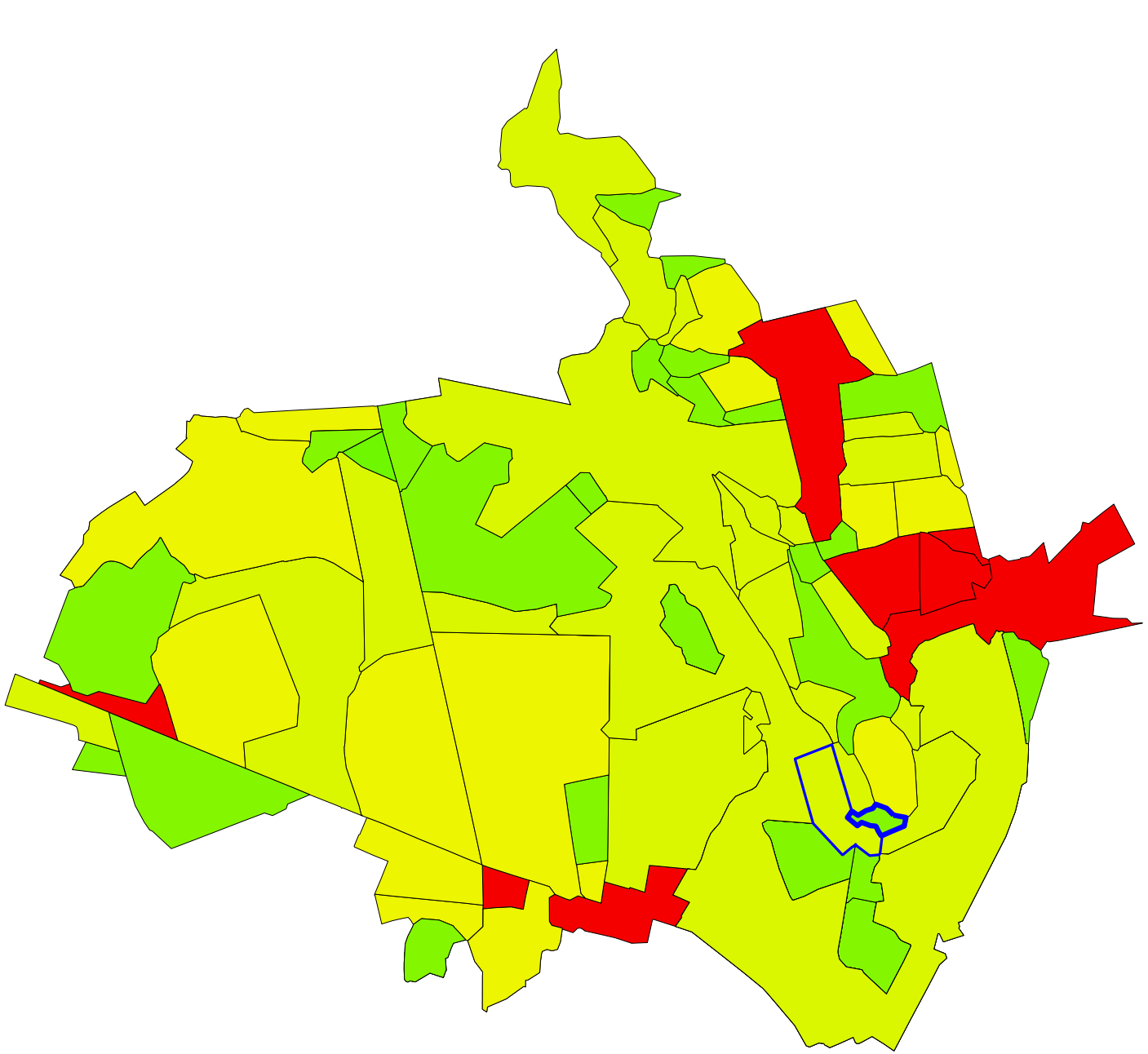


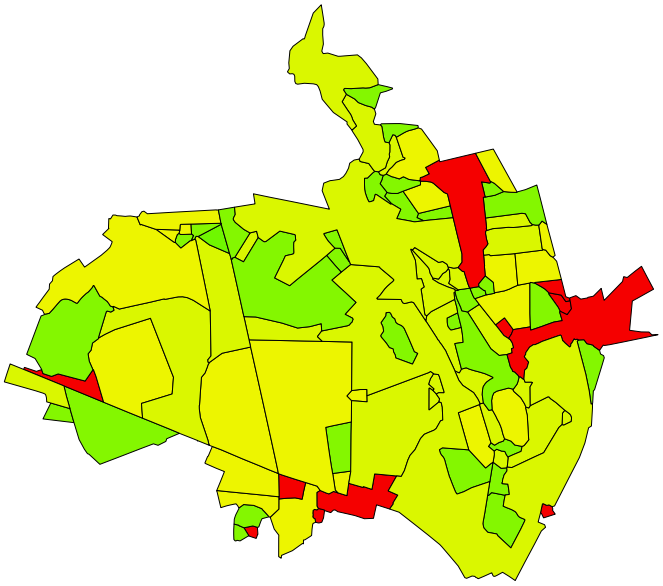


source: 92 patches

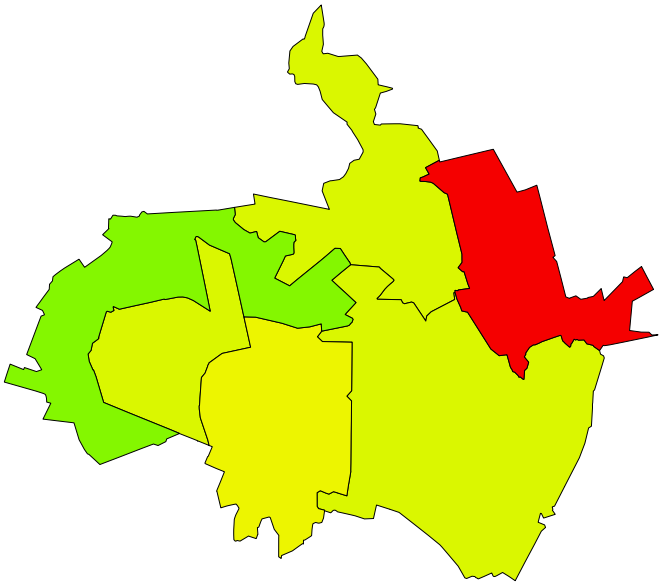


target: 6 patches

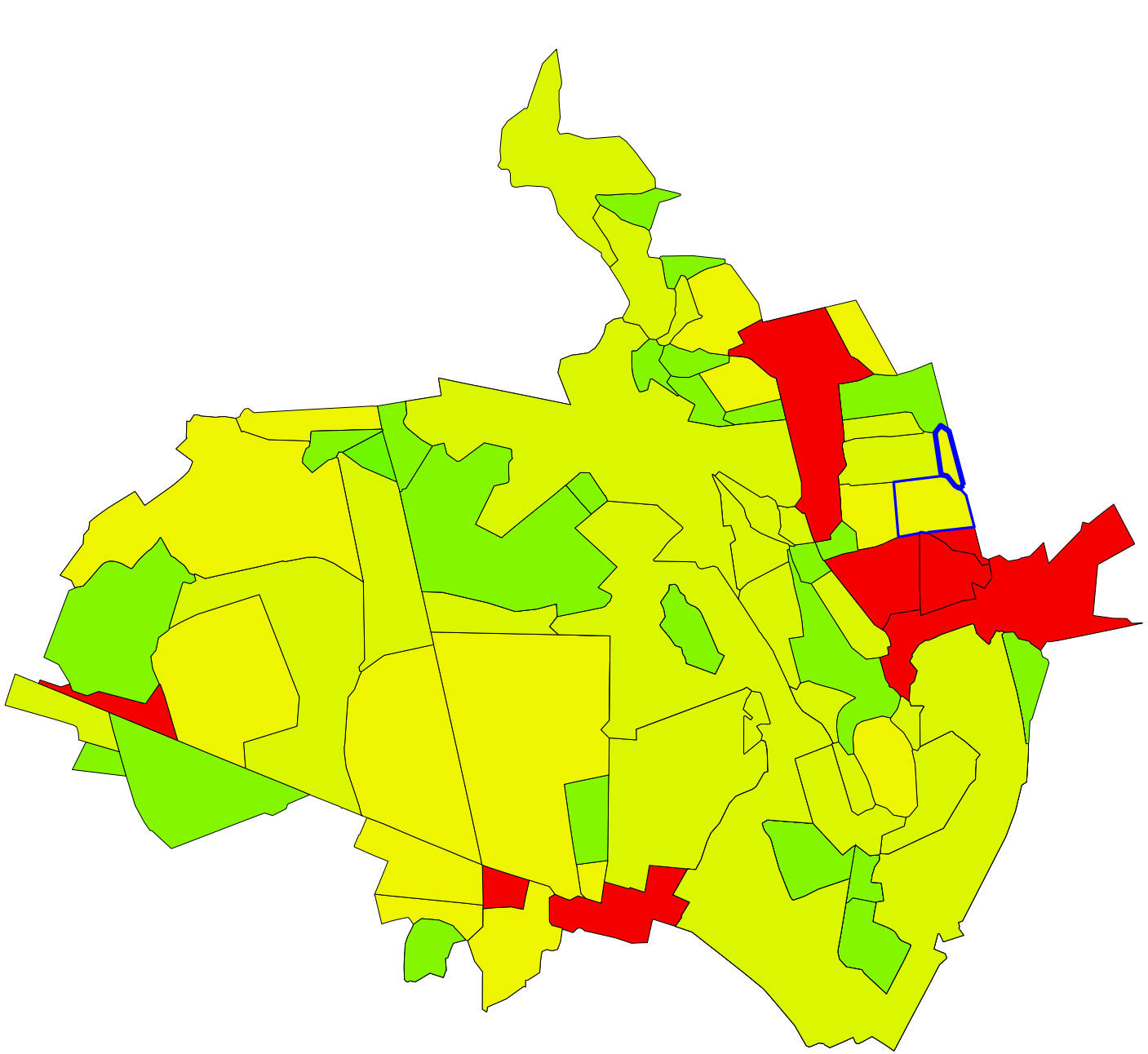


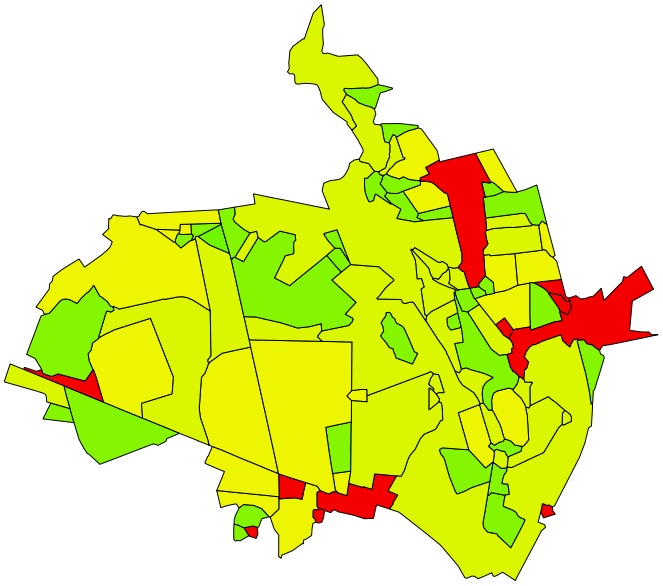


source: 92 patches

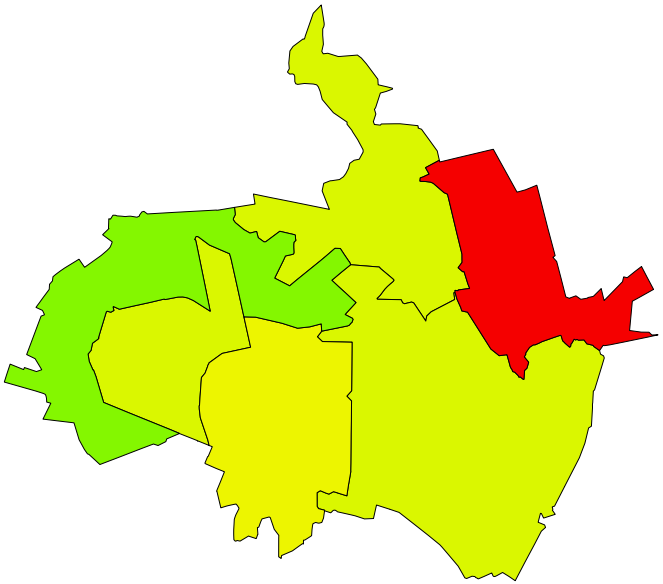


target: 6 patches

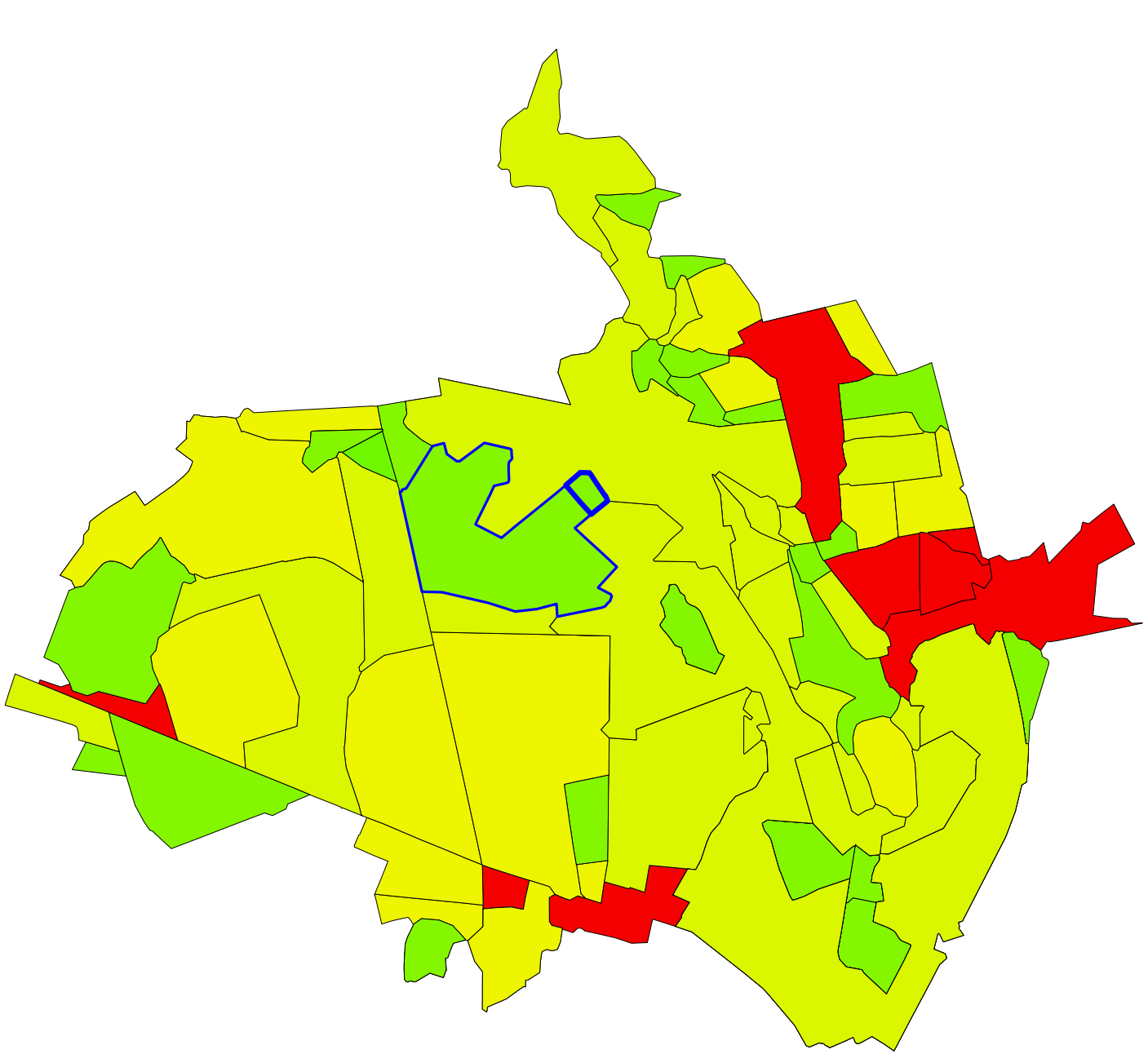


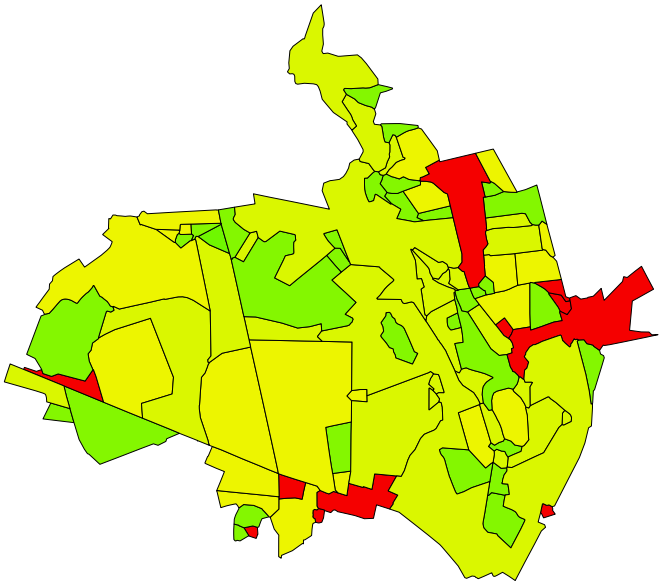


source: 92 patches

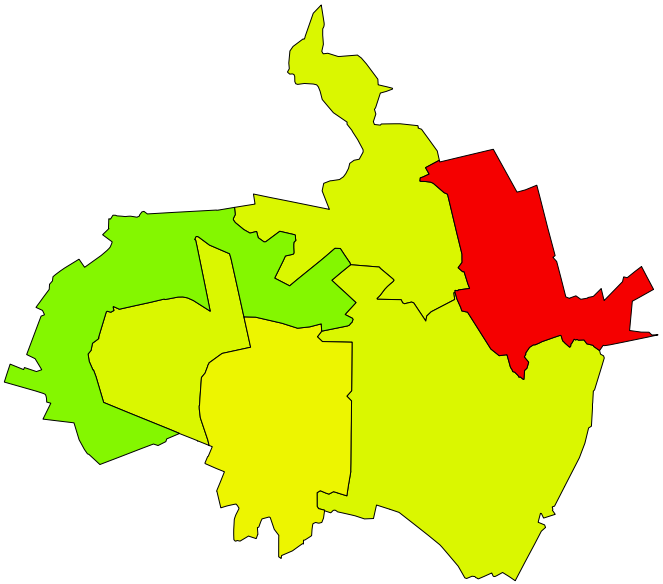


target: 6 patches

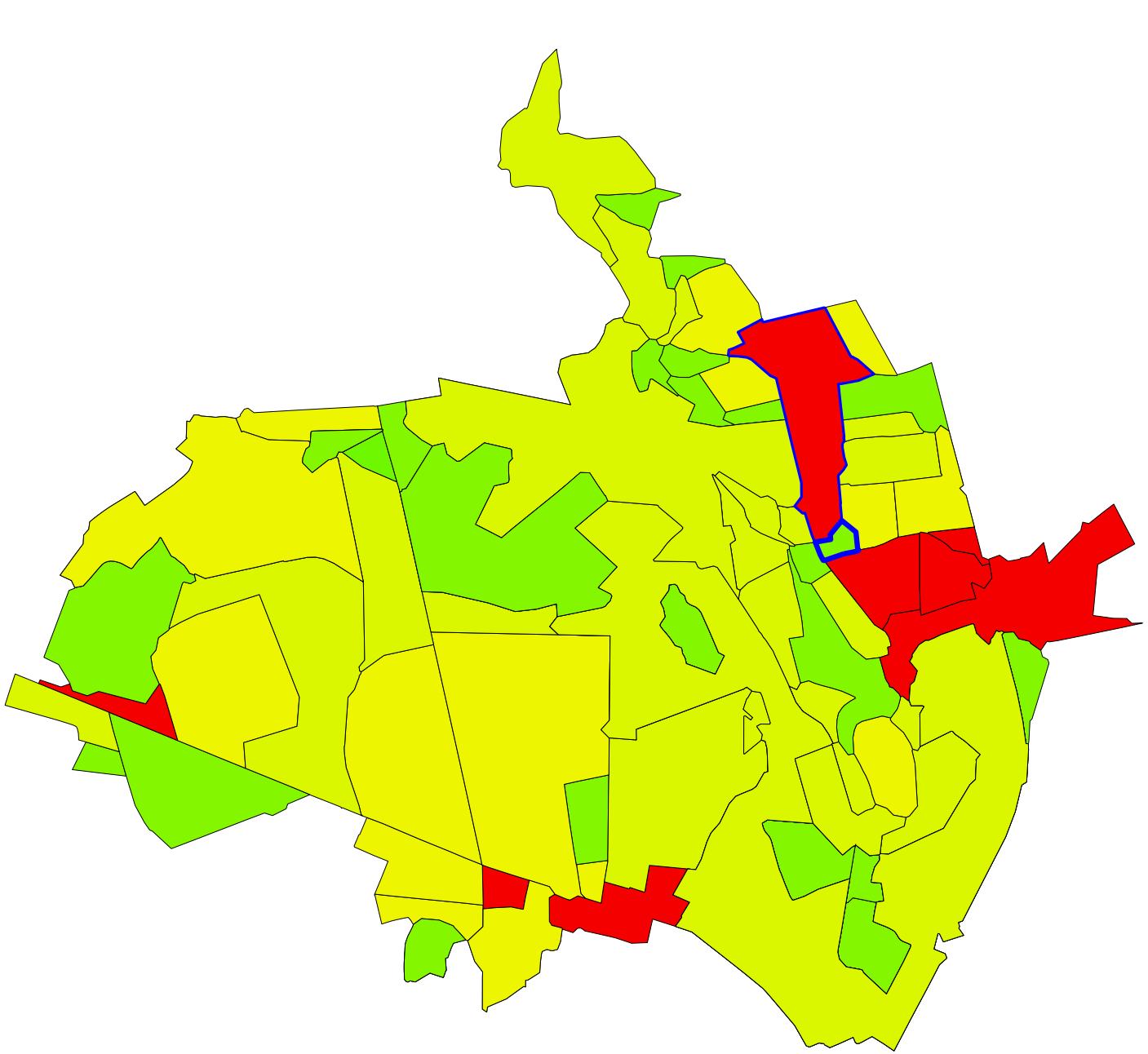


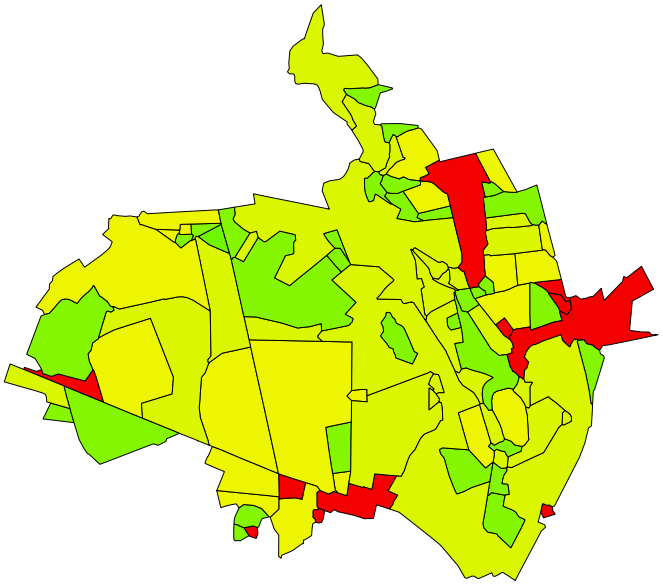


source: 92 patches

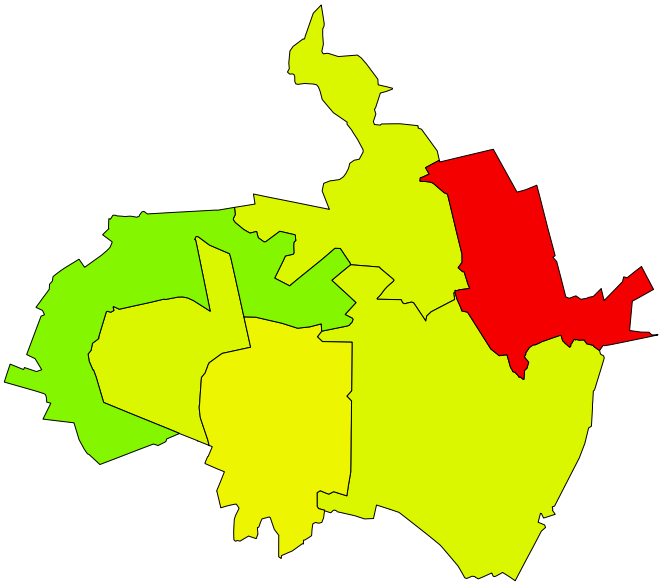


target: 6 patches

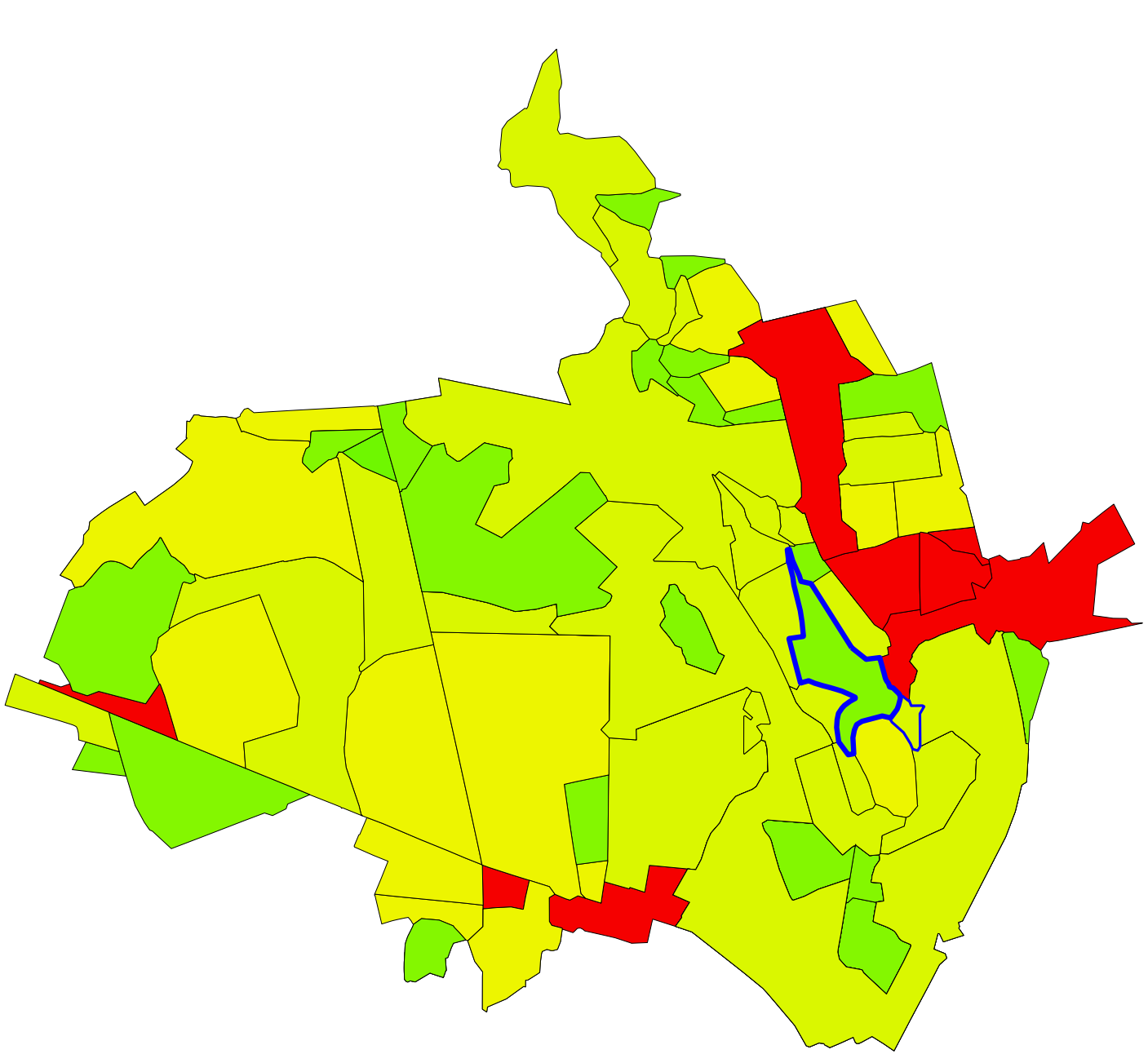


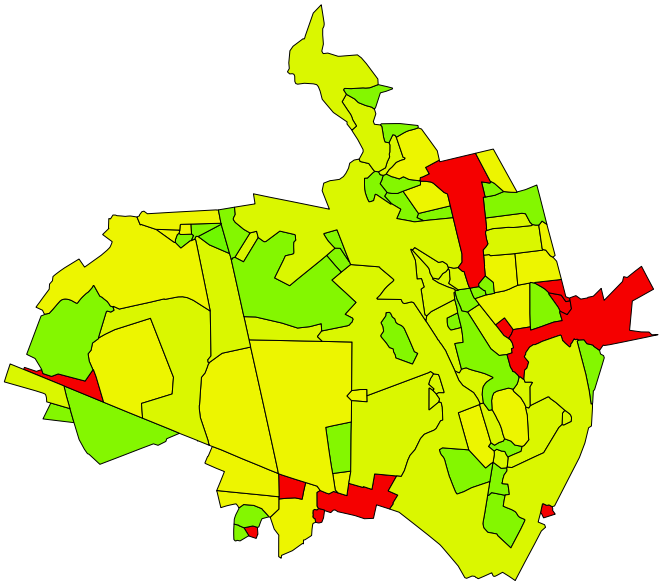


source: 92 patches

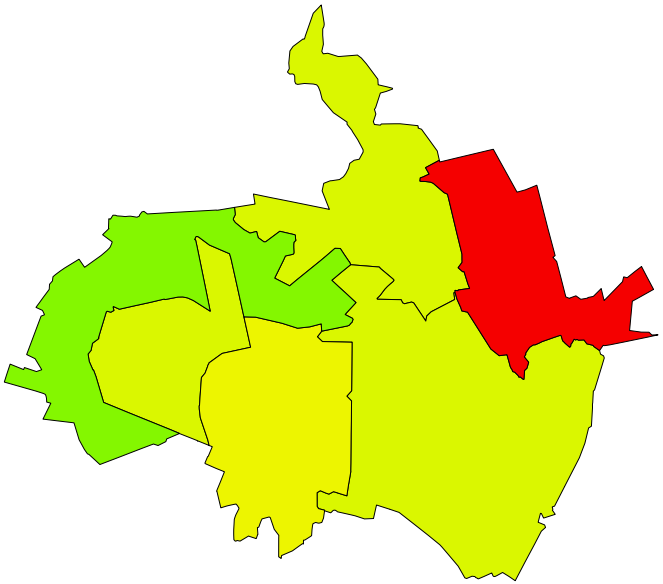


target: 6 patches

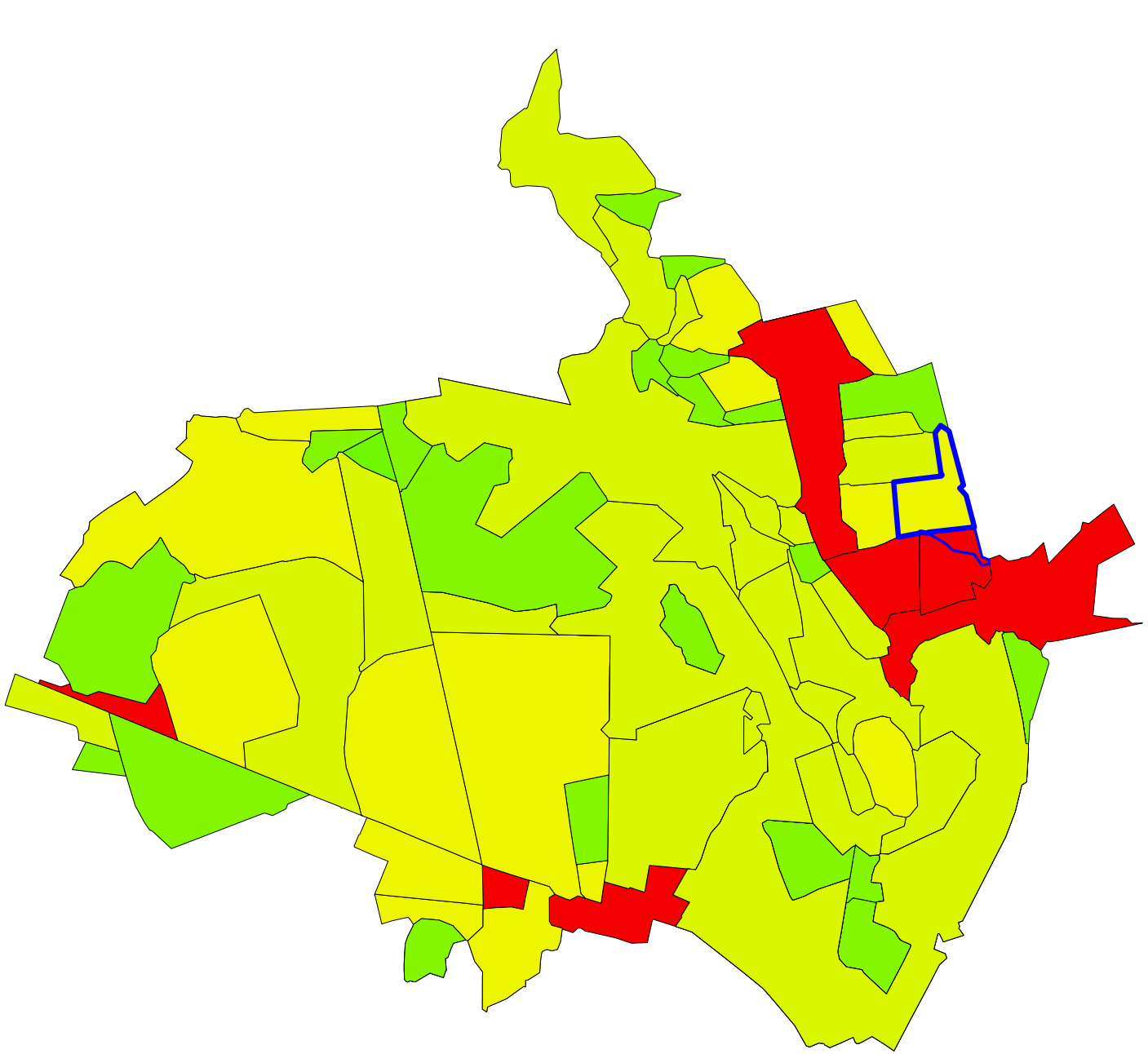


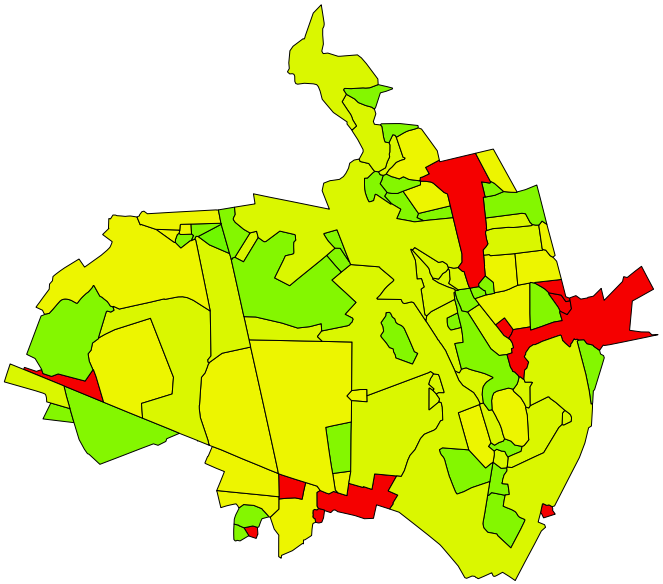


source: 92 patches

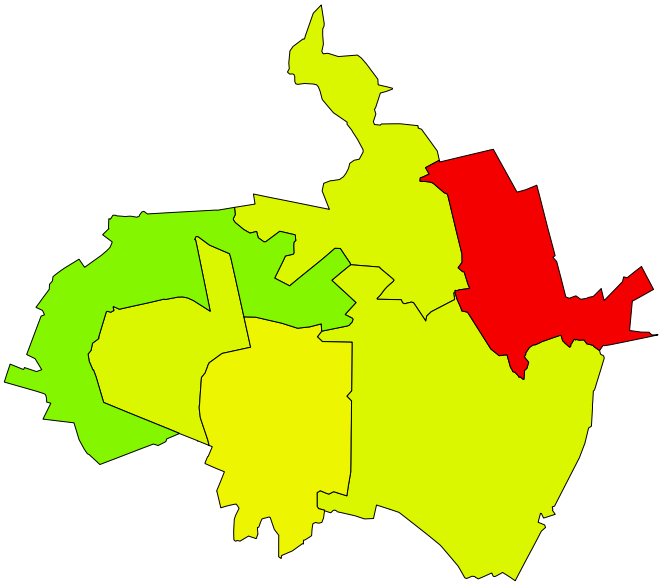


target: 6 patches

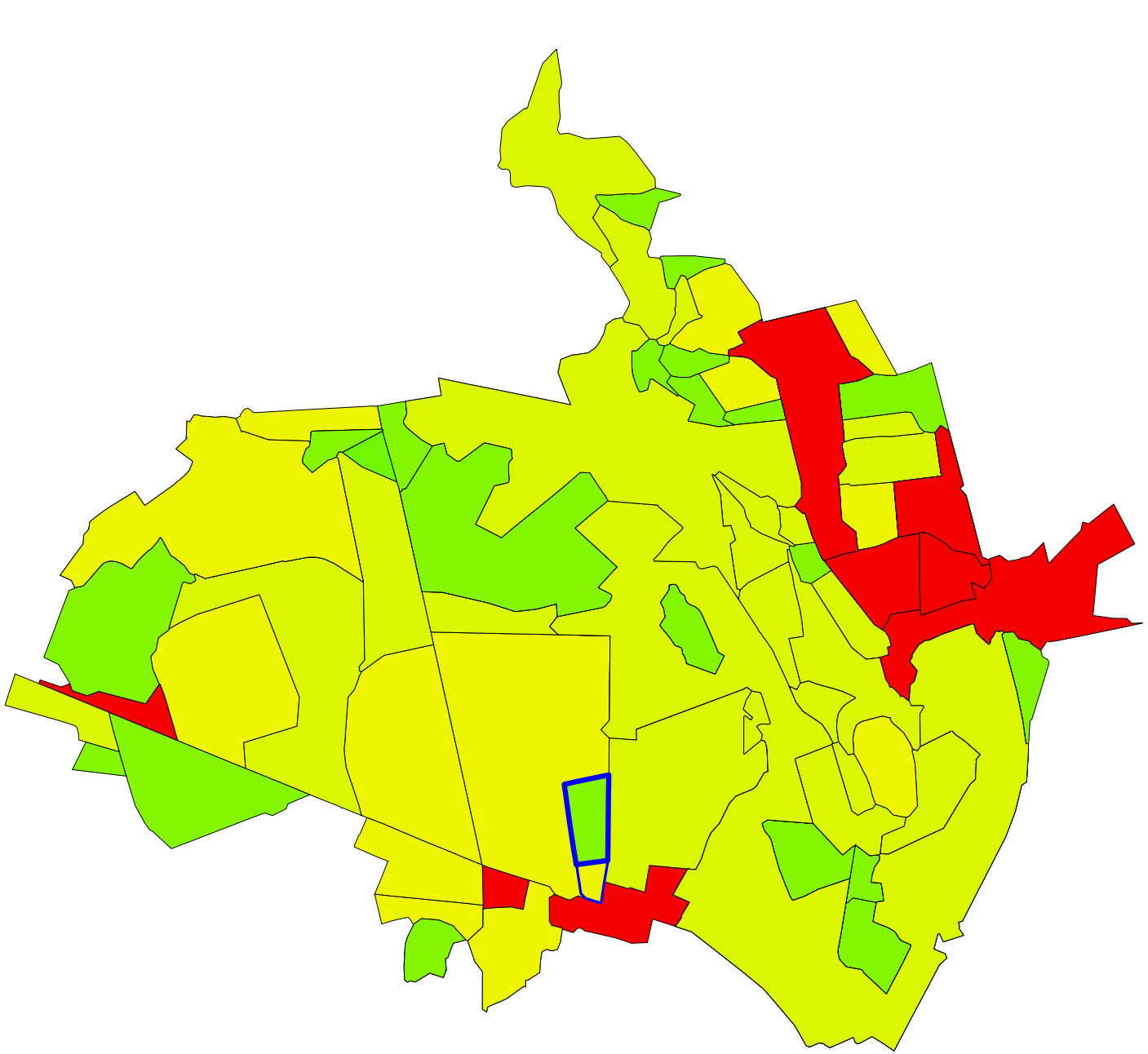


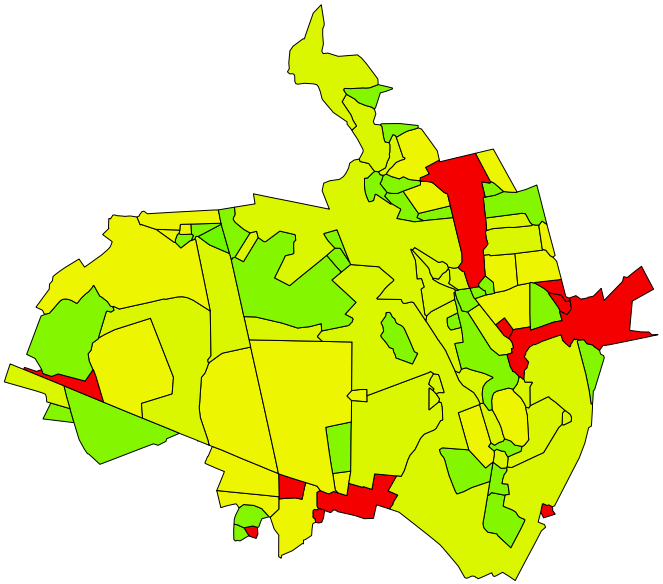


source: 92 patches

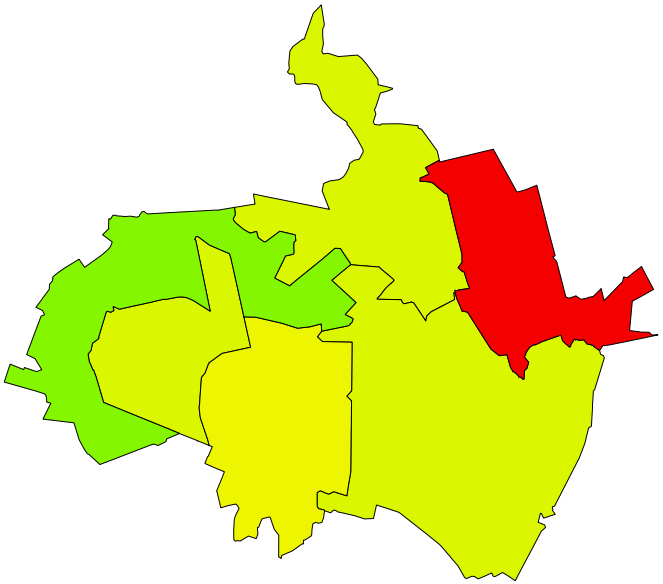


target: 6 patches

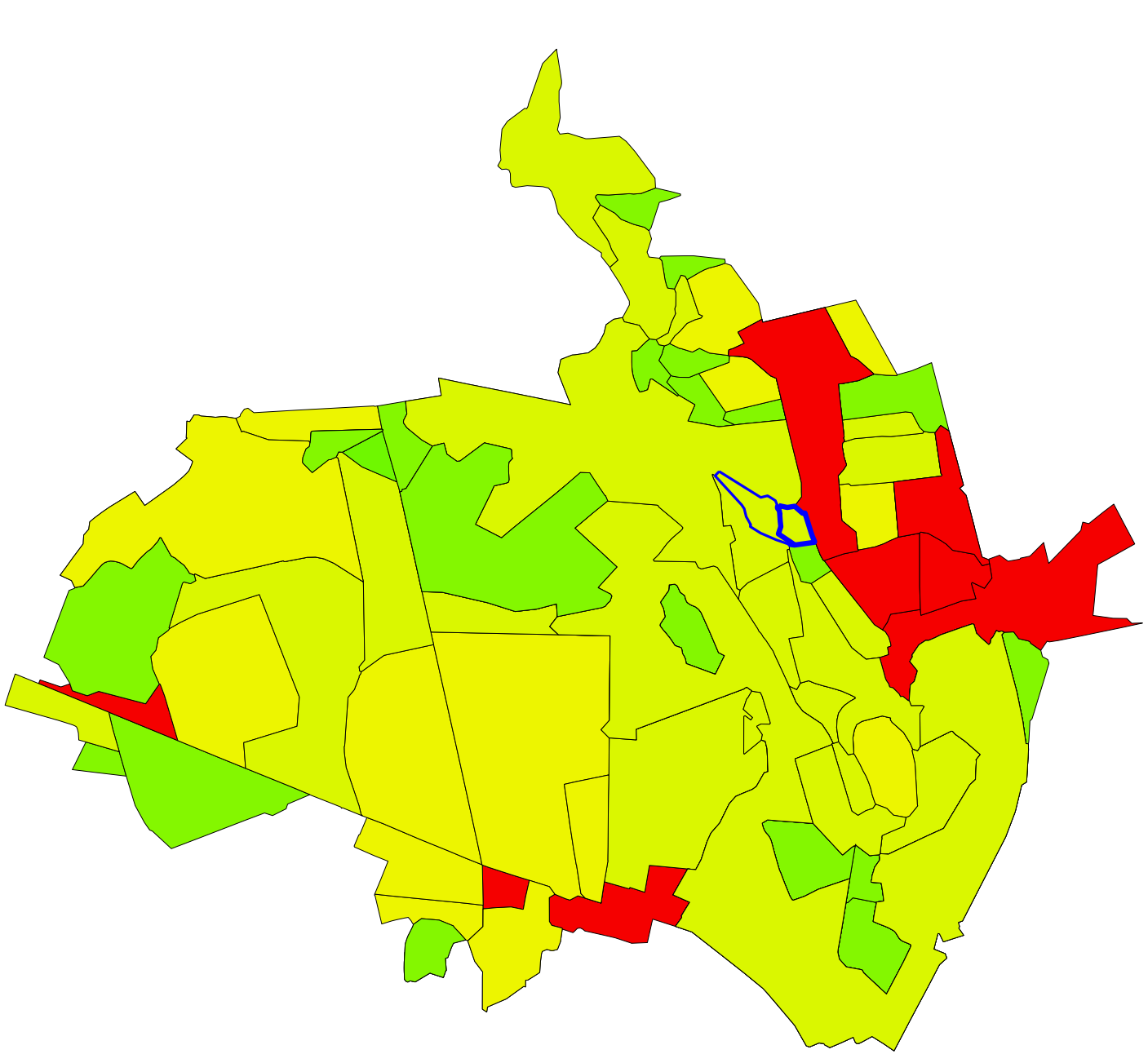


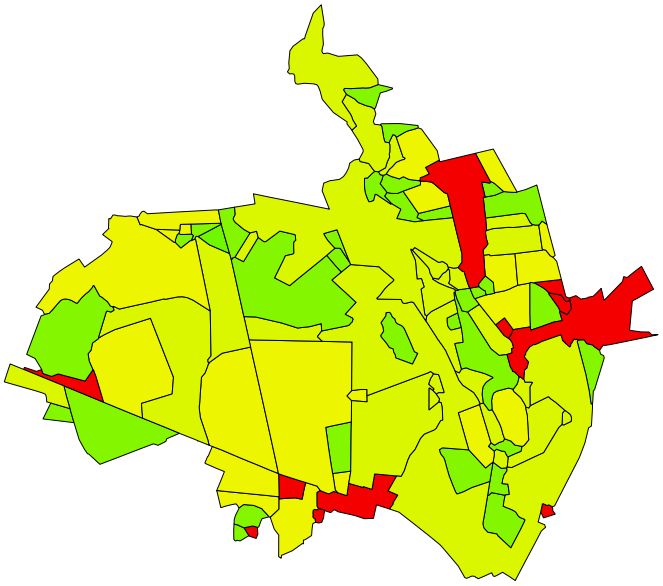


source: 92 patches

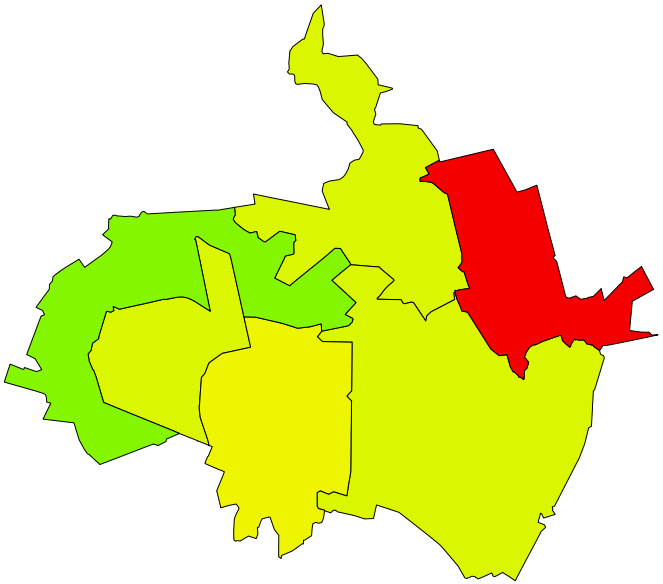


target: 6 patches

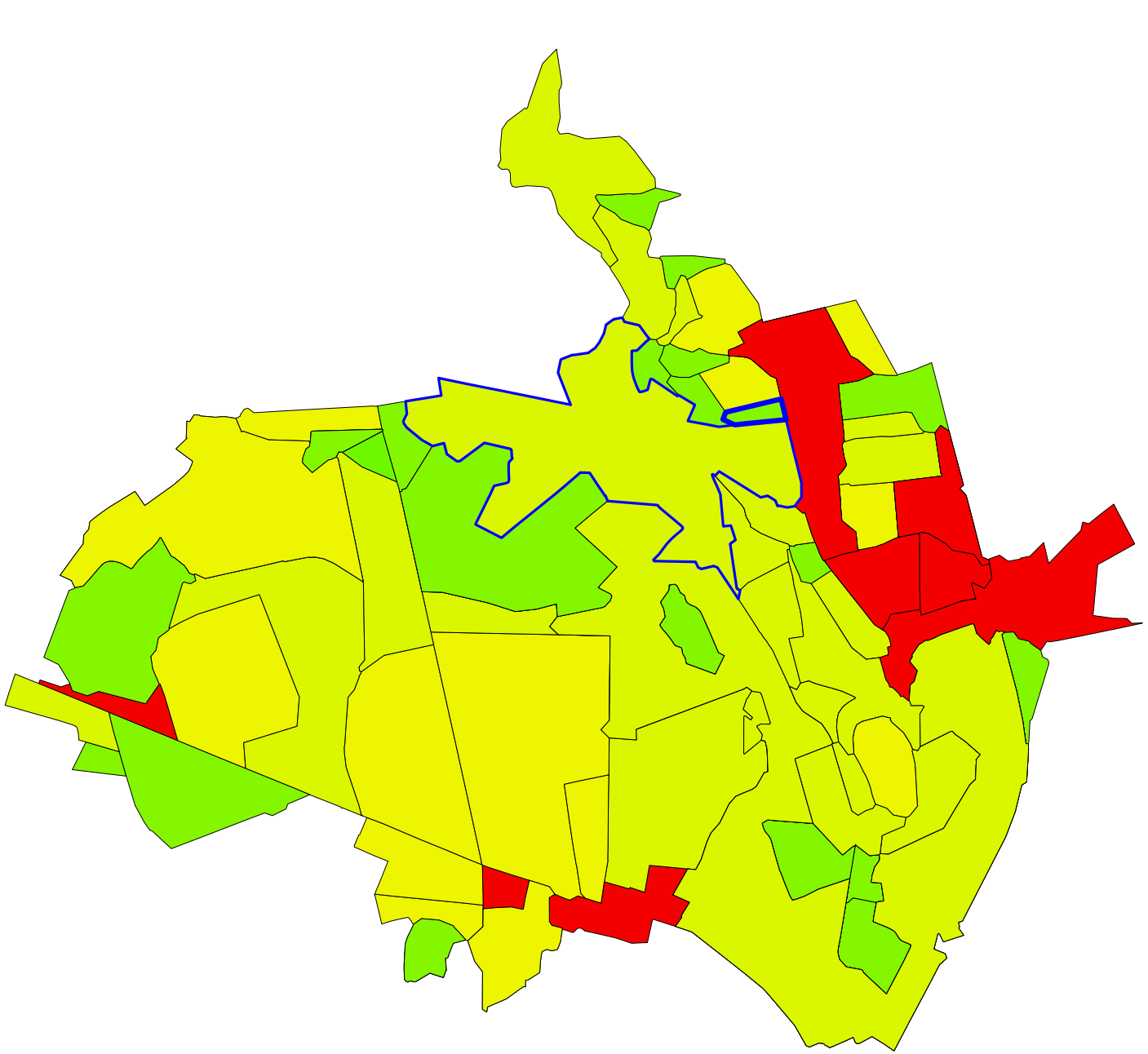


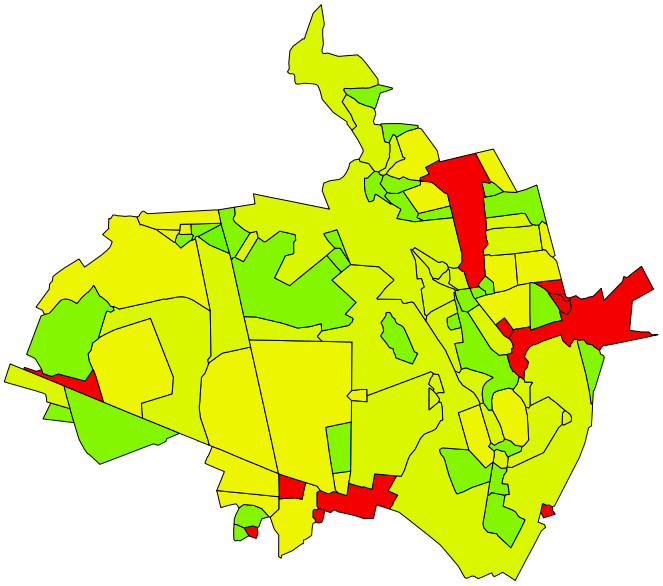


source: 92 patches

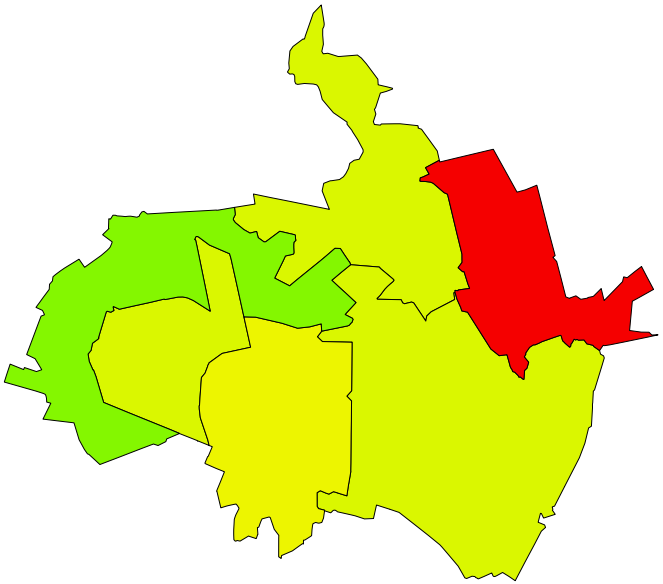


target: 6 patches

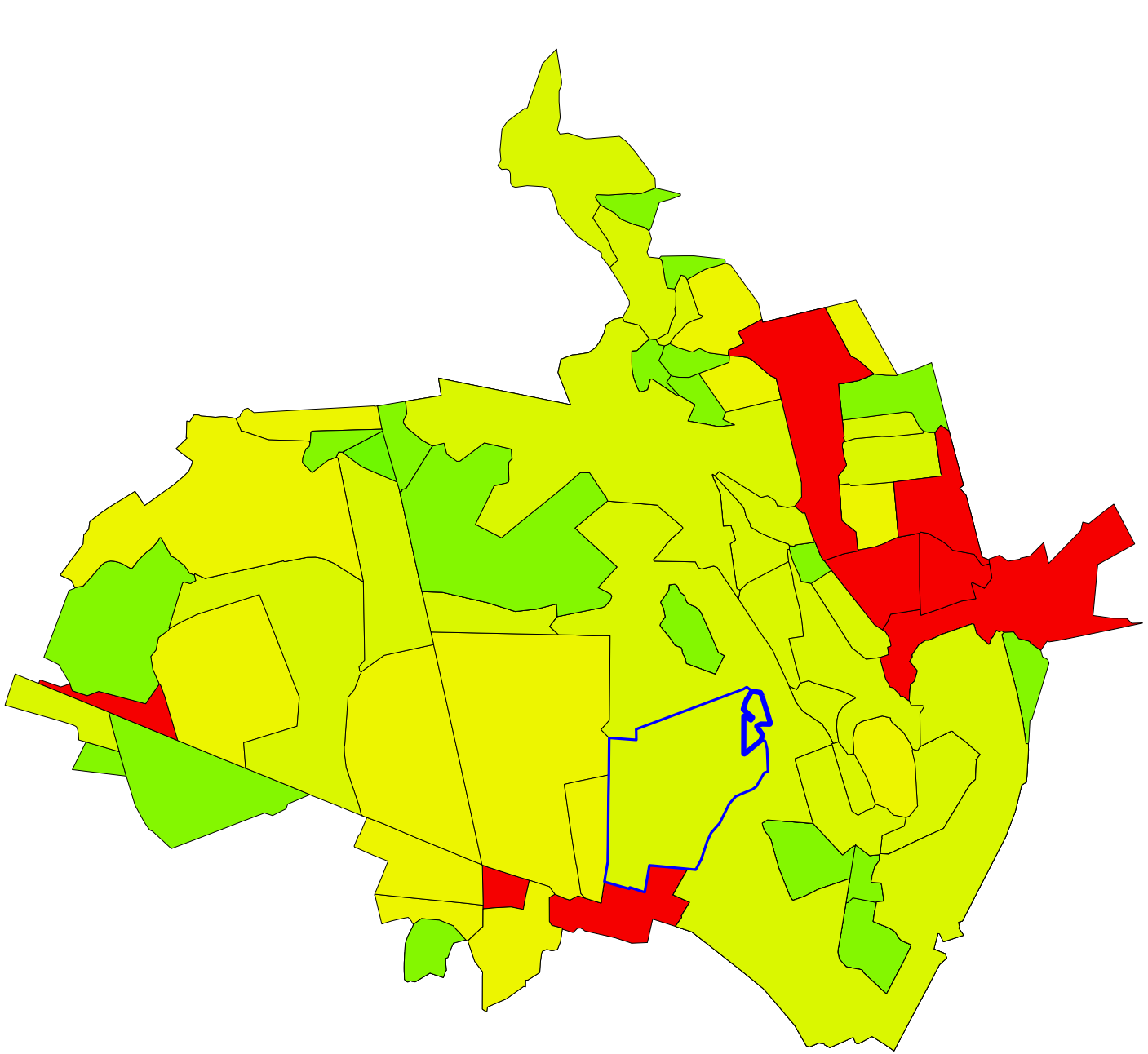


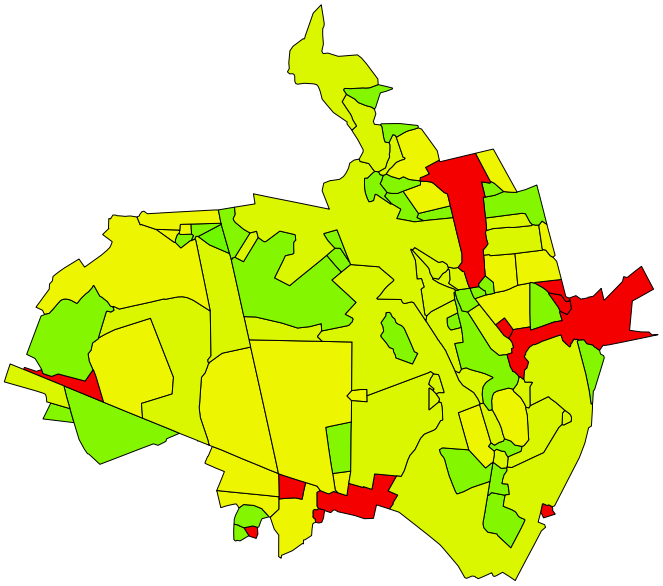


source: 92 patches

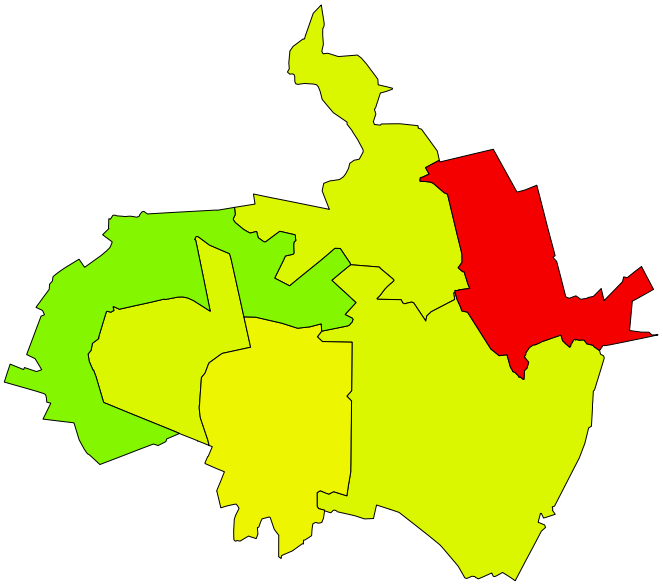


target: 6 patches

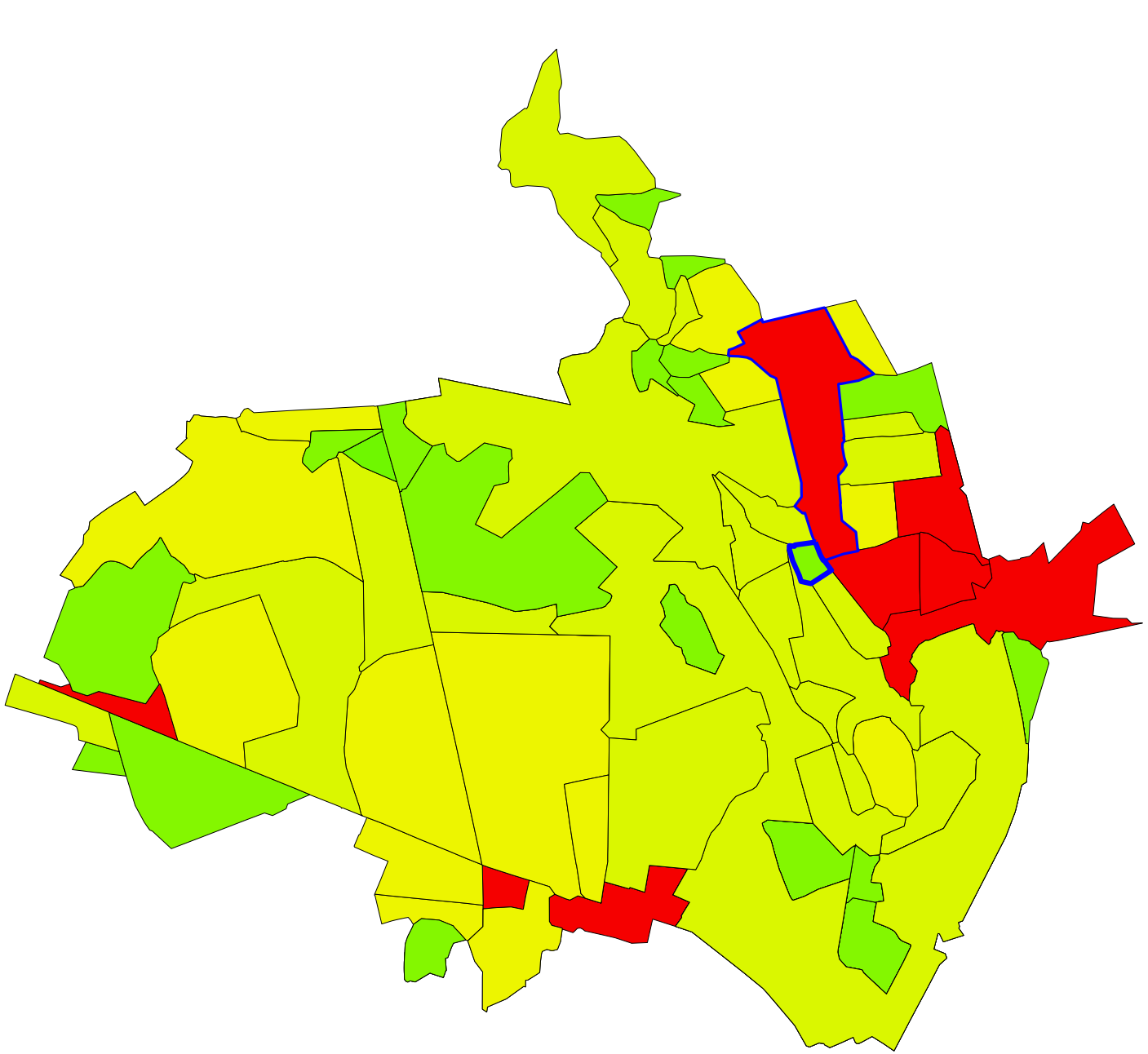


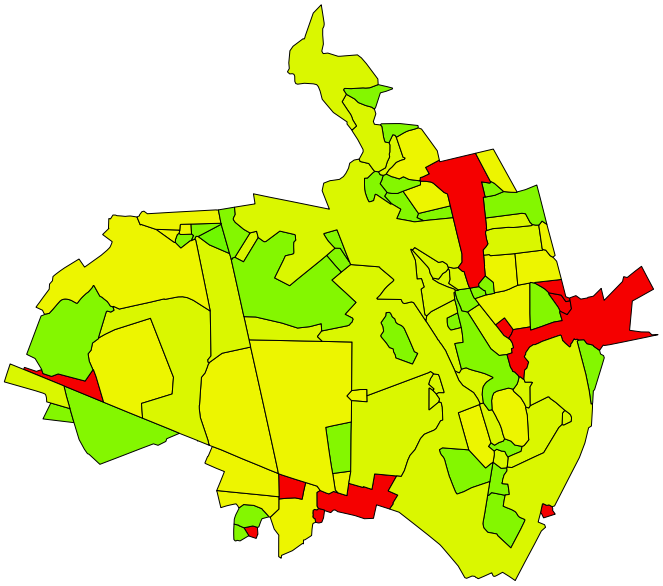


source: 92 patches

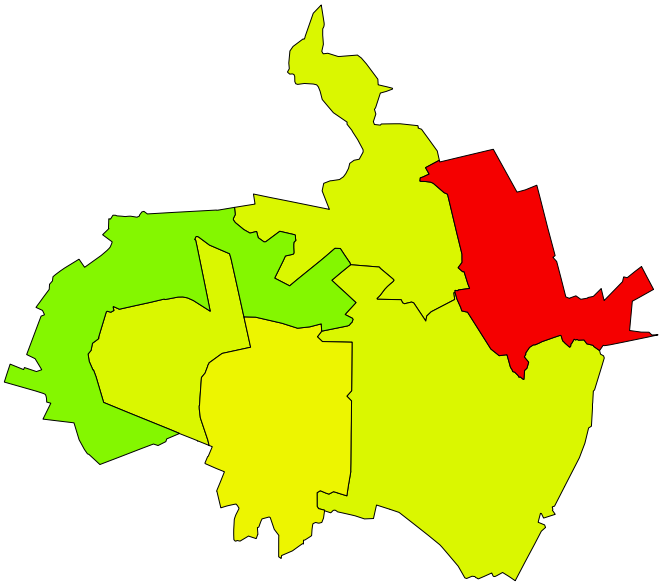


target: 6 patches

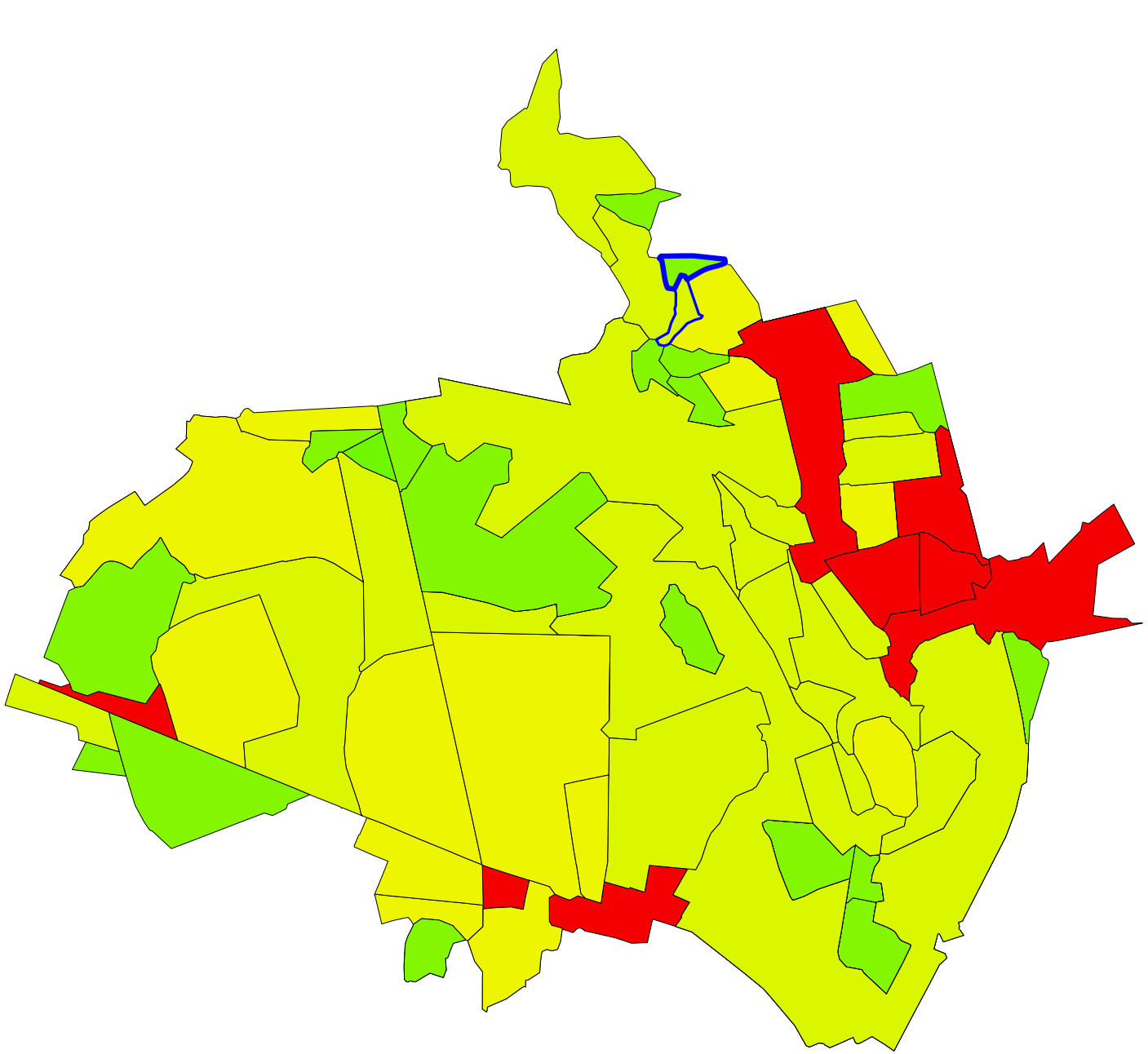


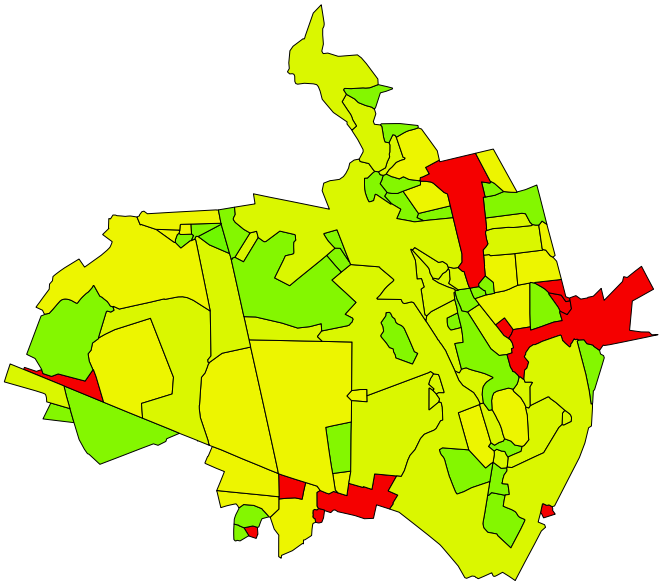


source: 92 patches

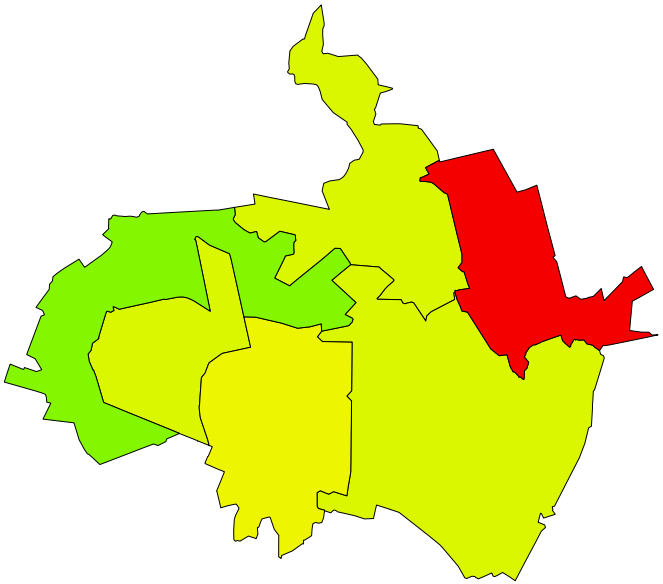


target: 6 patches

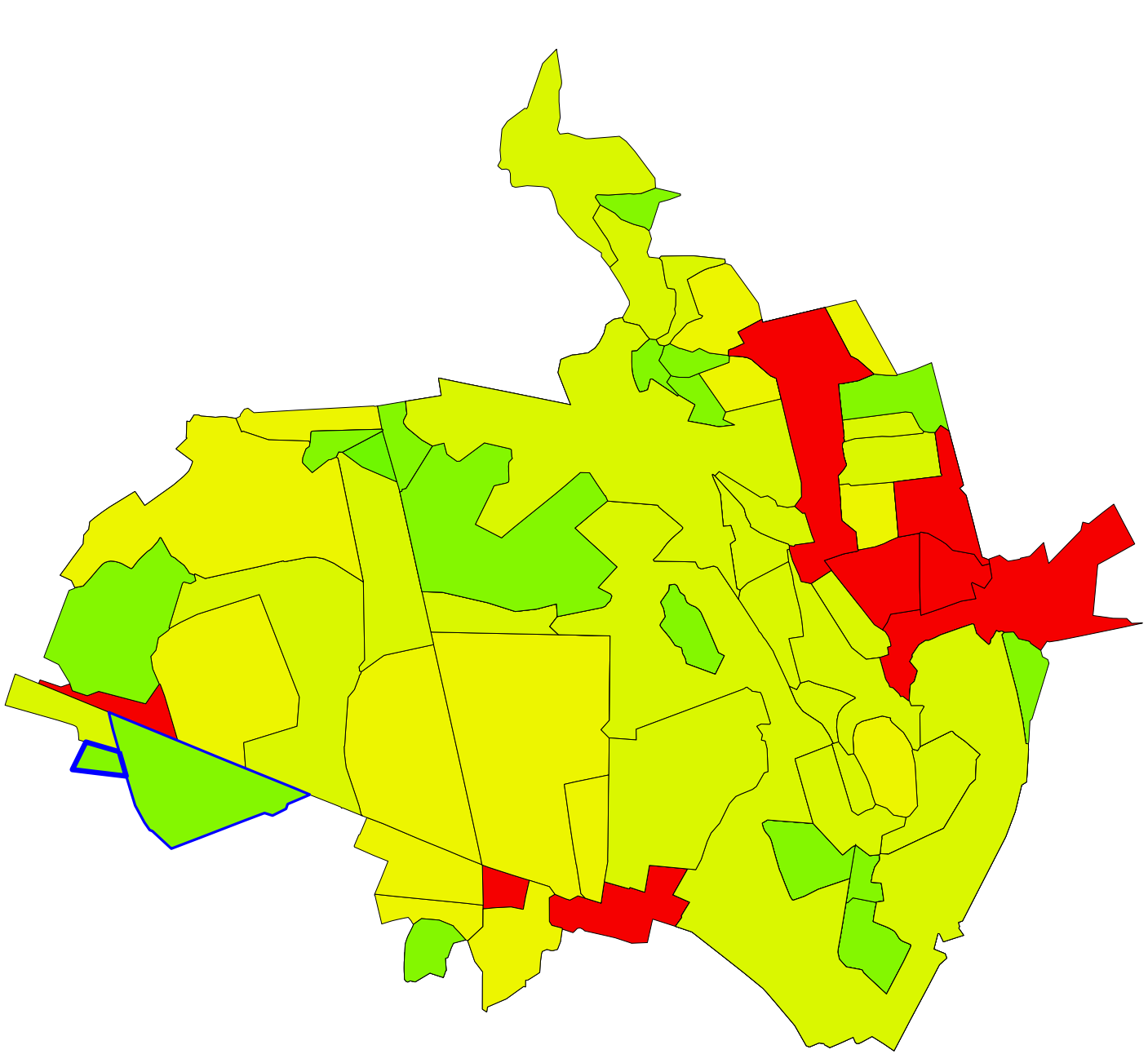


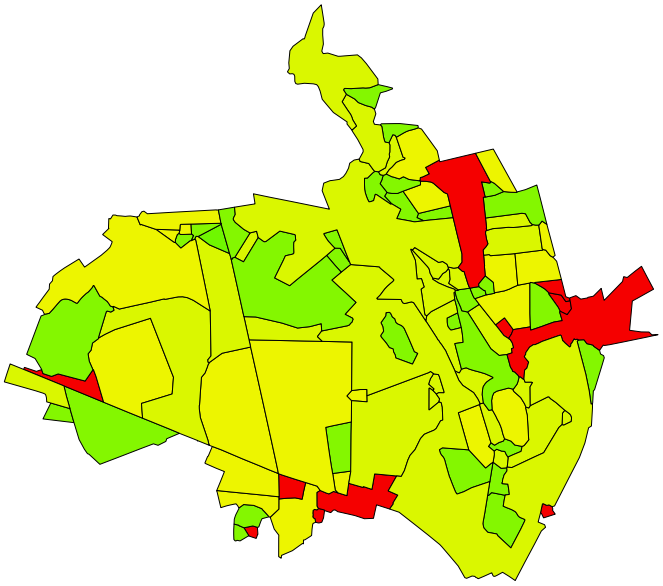


source: 92 patches

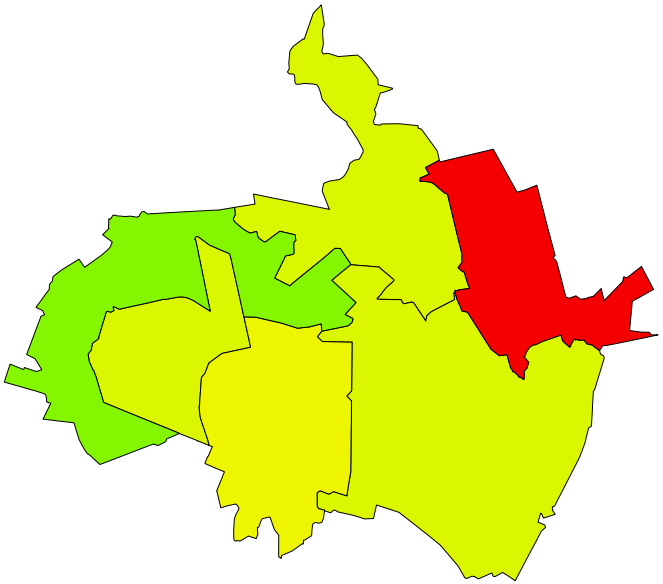


target: 6 patches

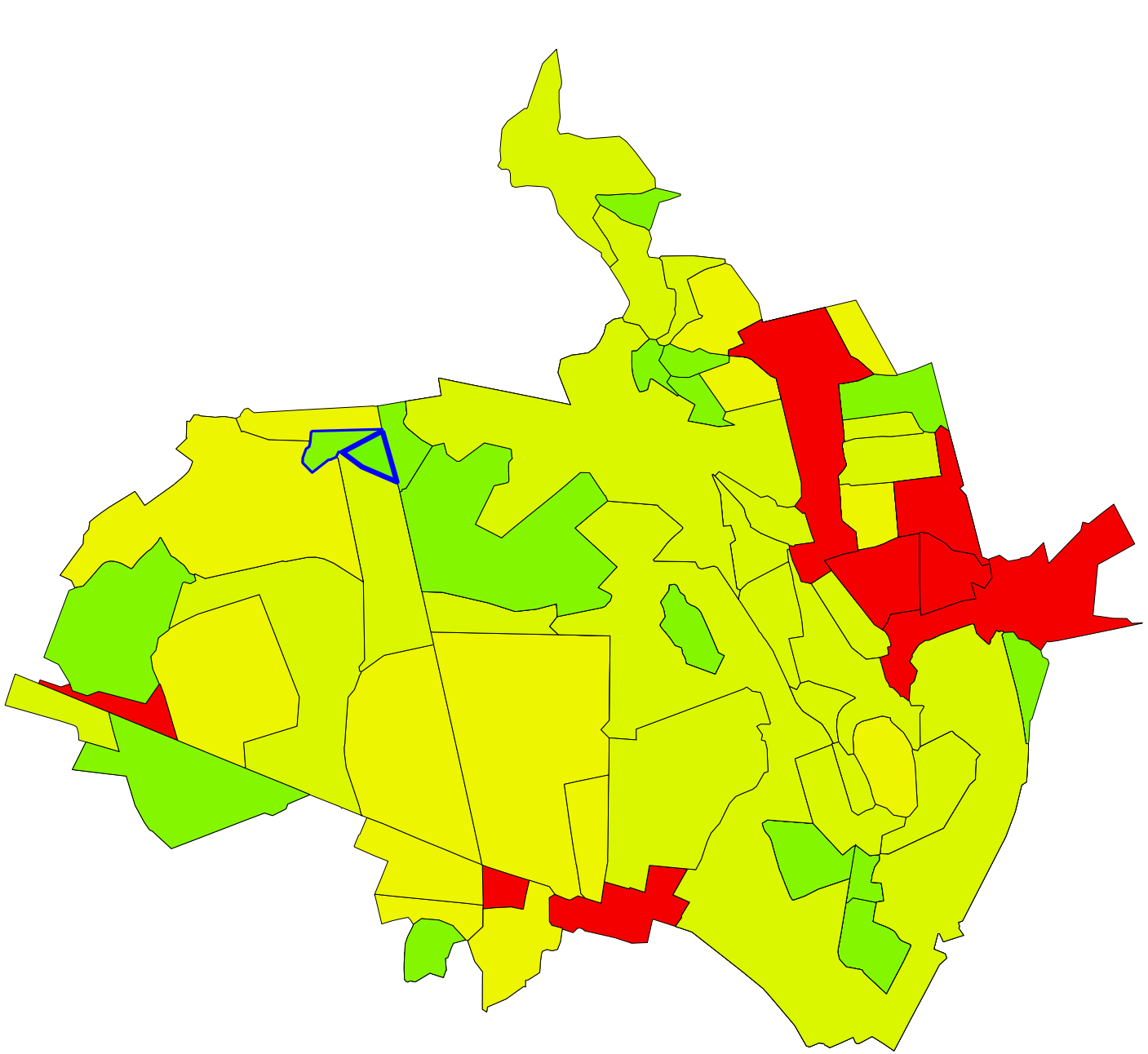


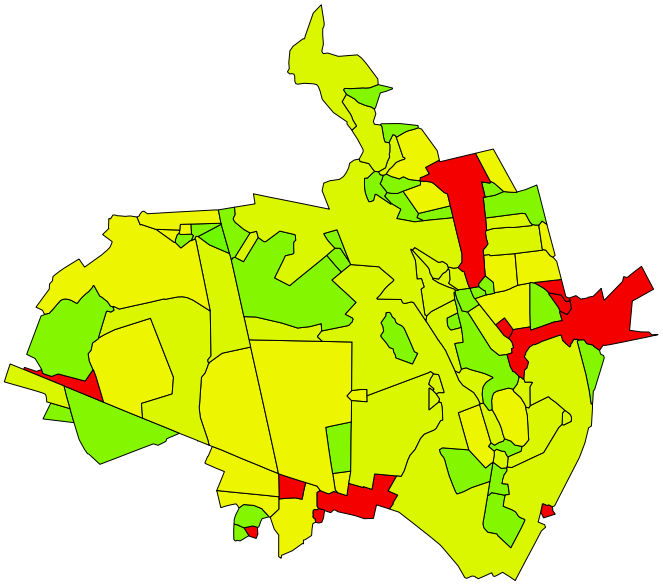


source: 92 patches

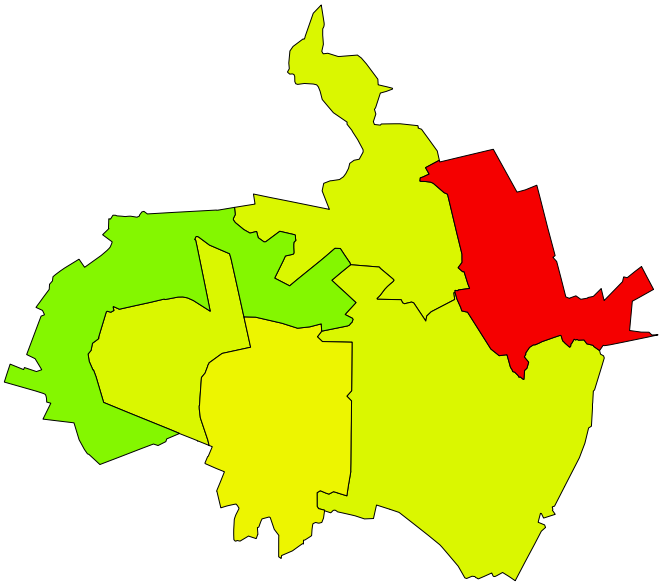


target: 6 patches

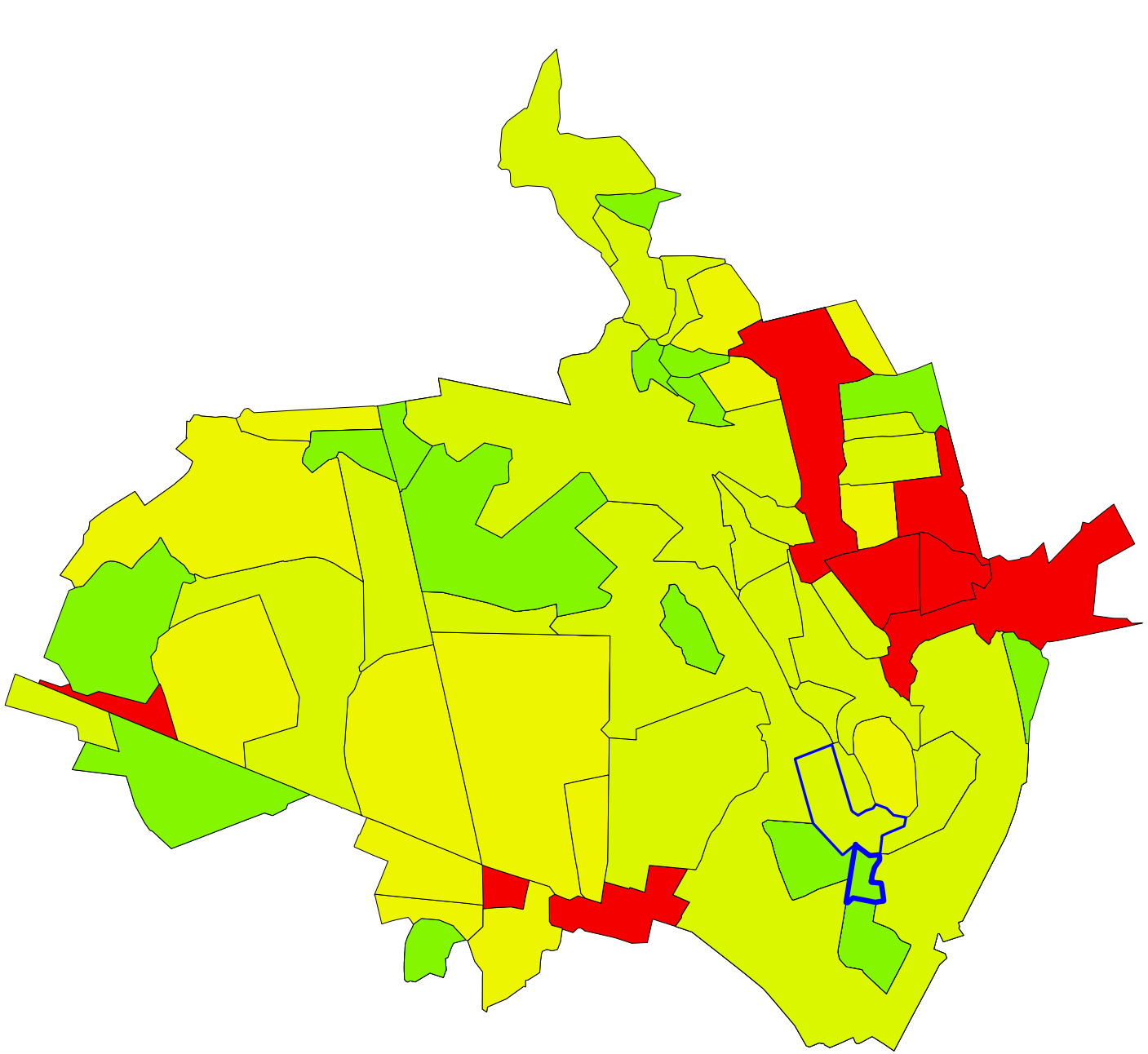


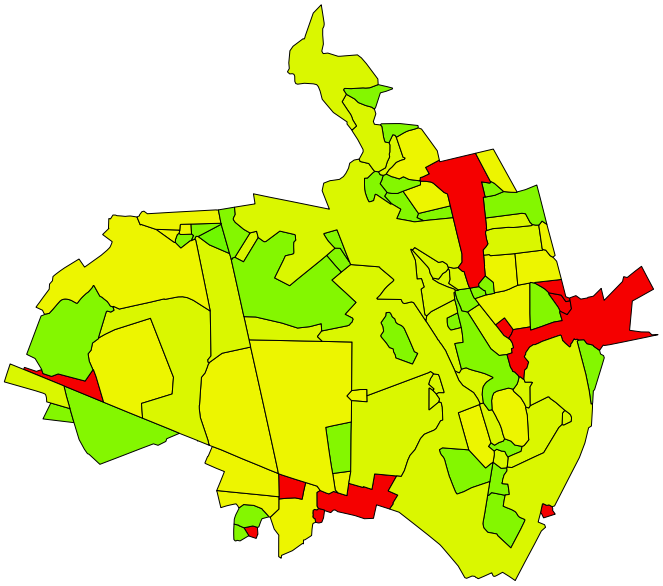


source: 92 patches

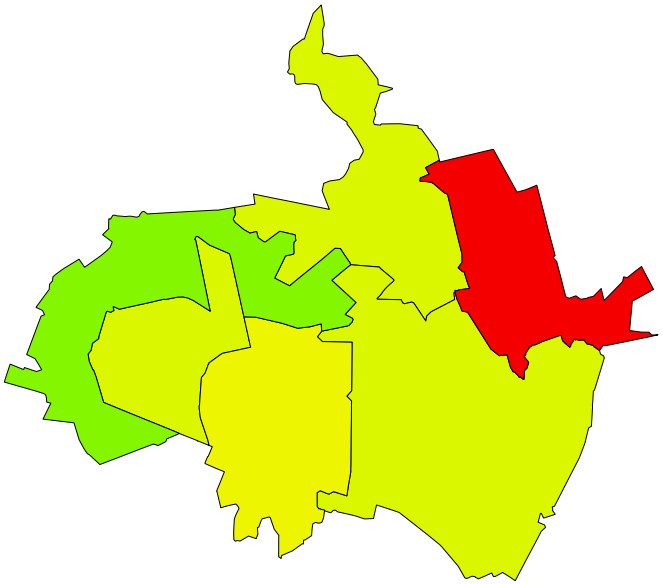


target: 6 patches

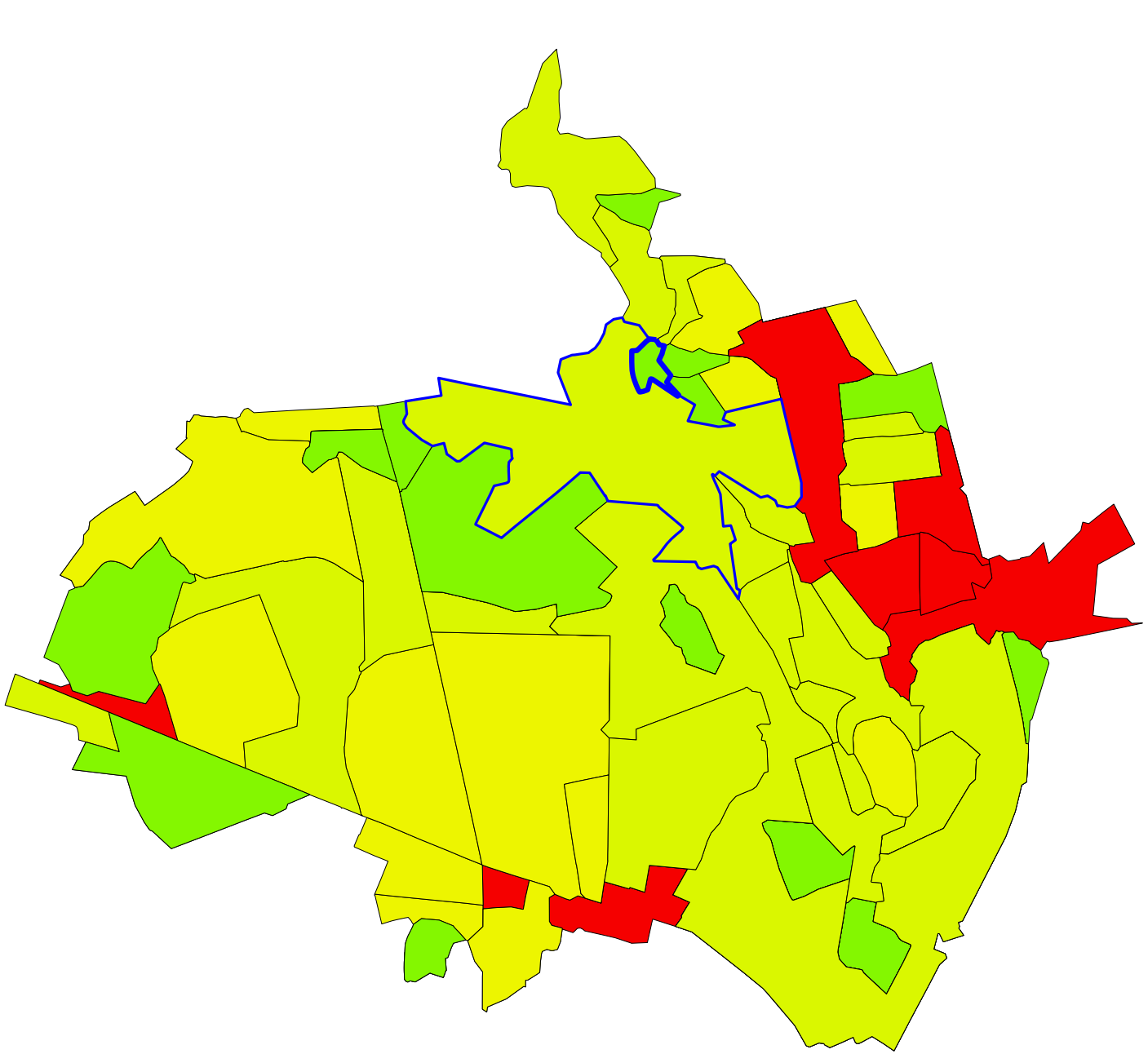


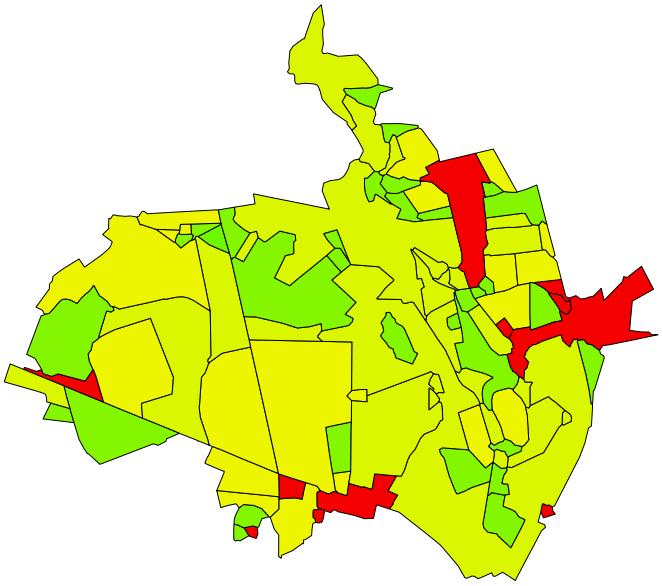


source: 92 patches

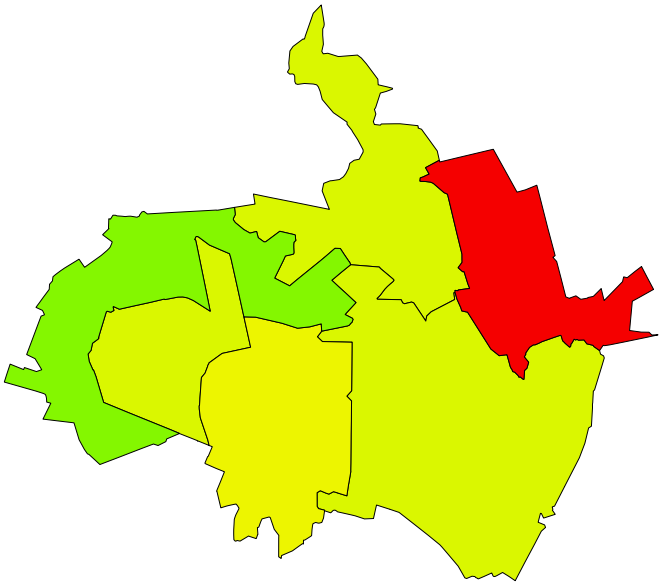


target: 6 patches

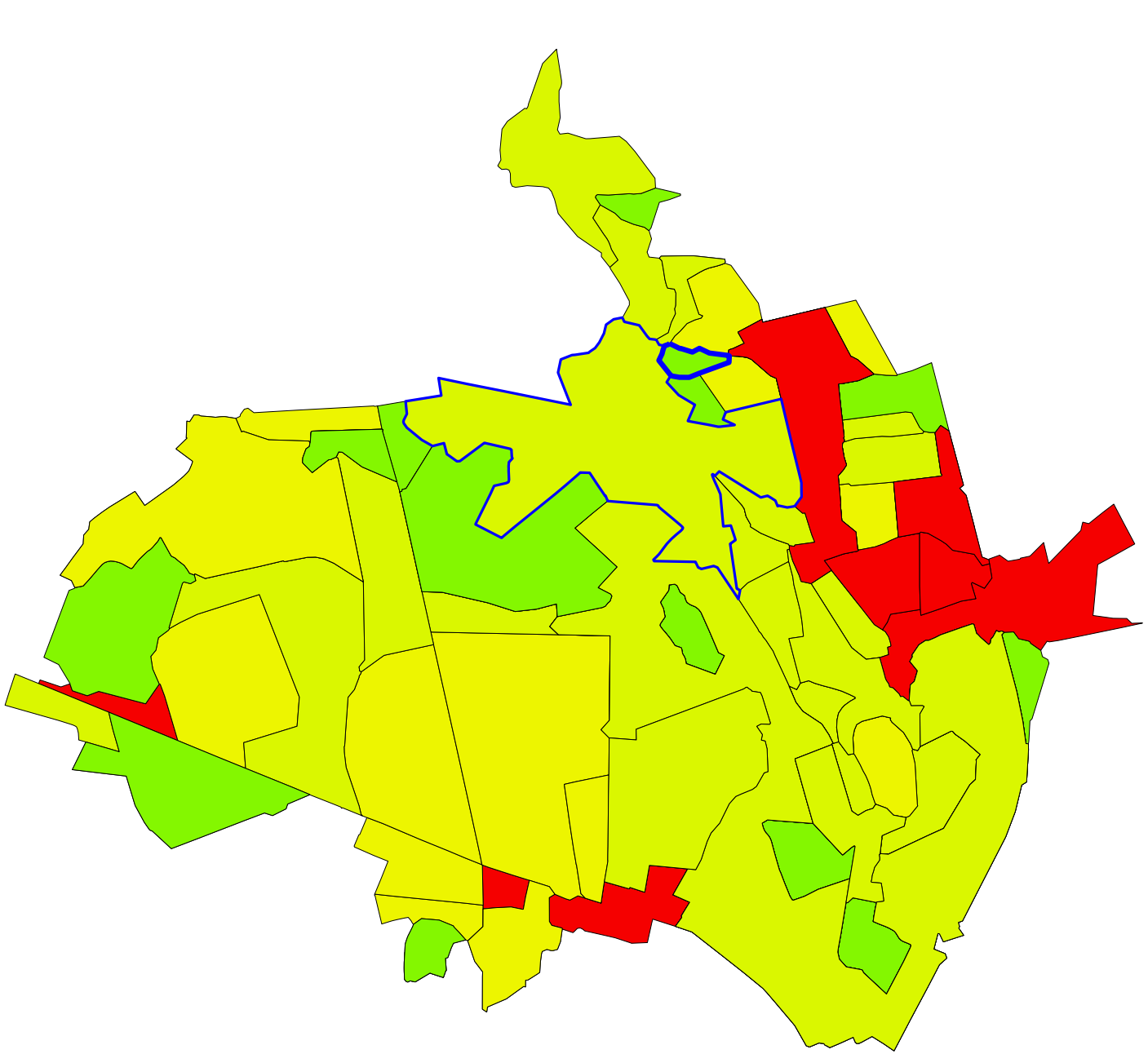


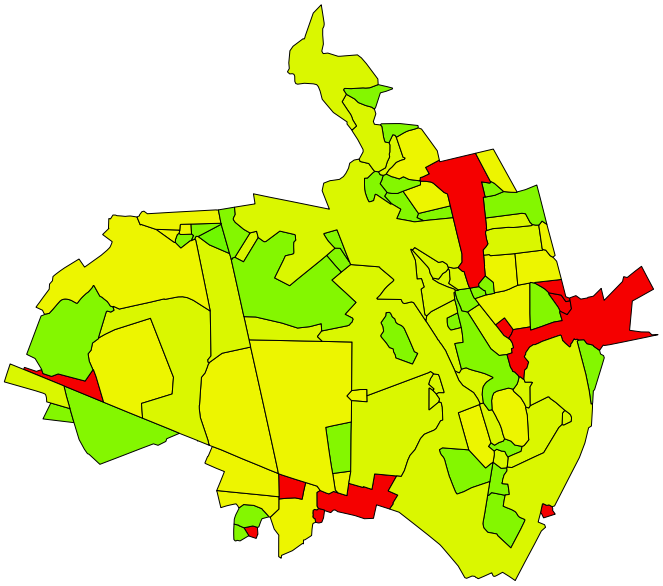


source: 92 patches

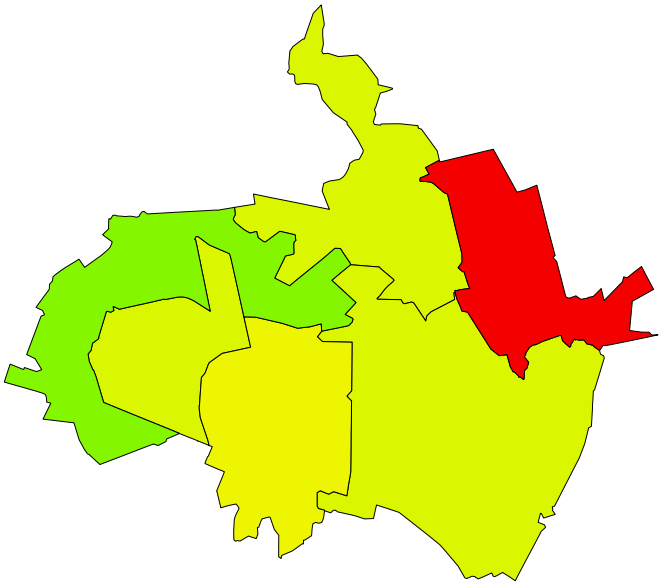


target: 6 patches

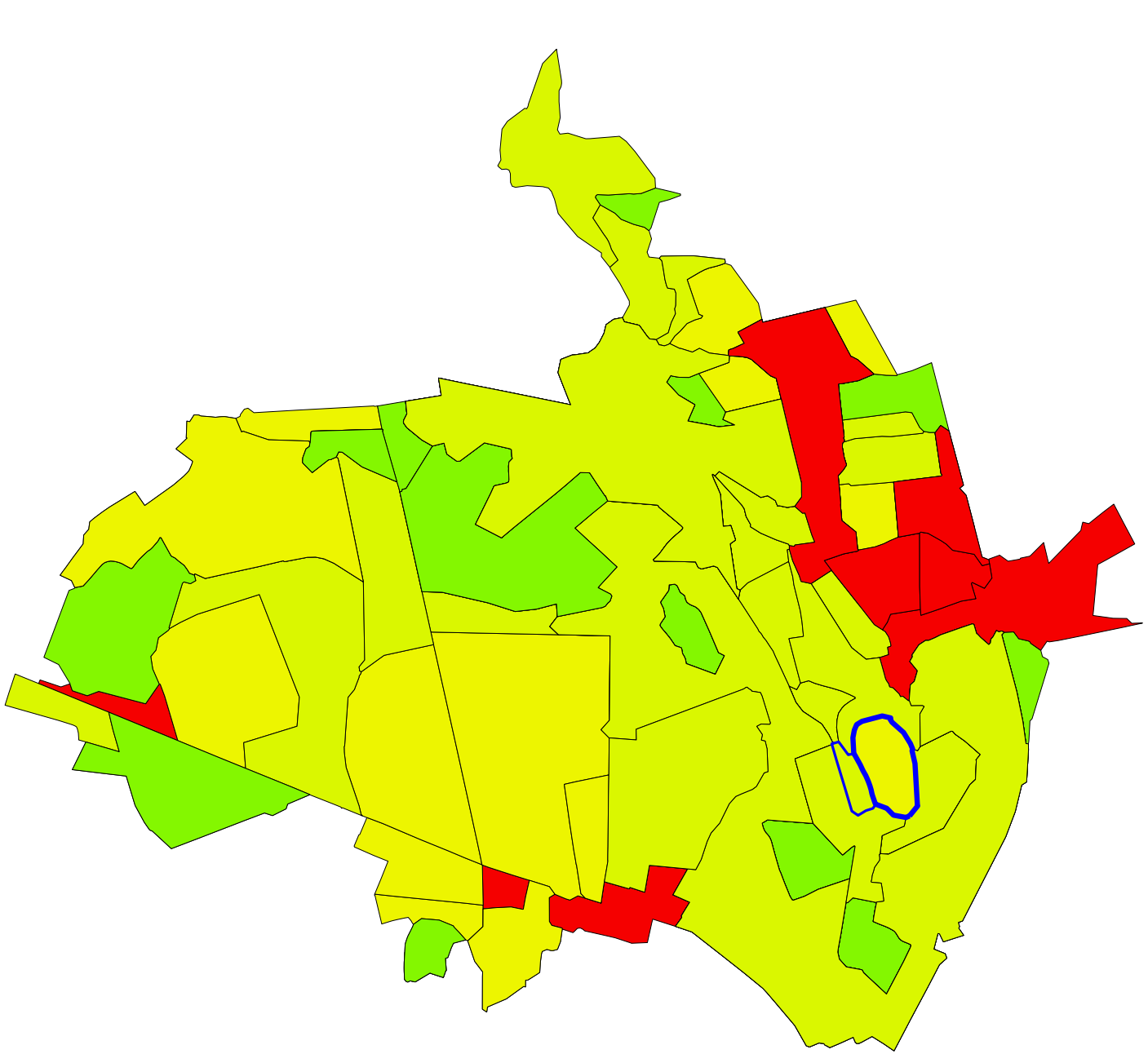


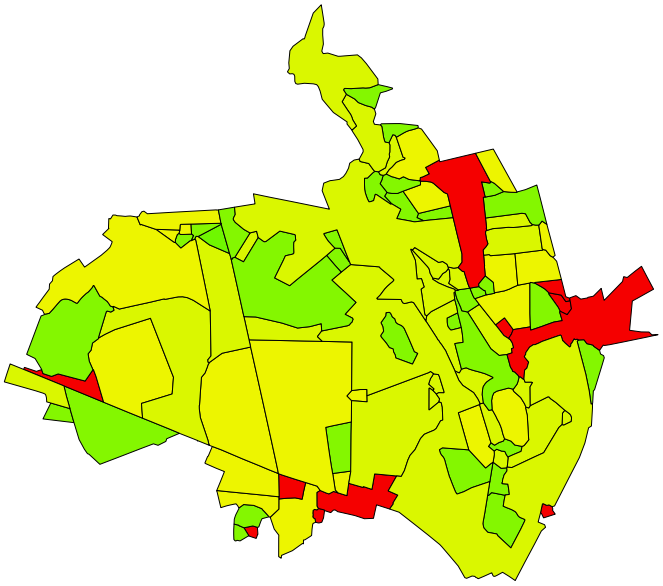


source: 92 patches

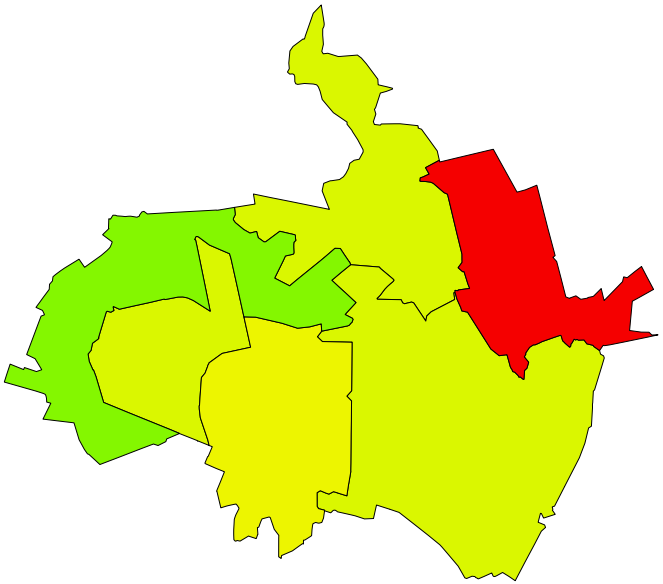


target: 6 patches

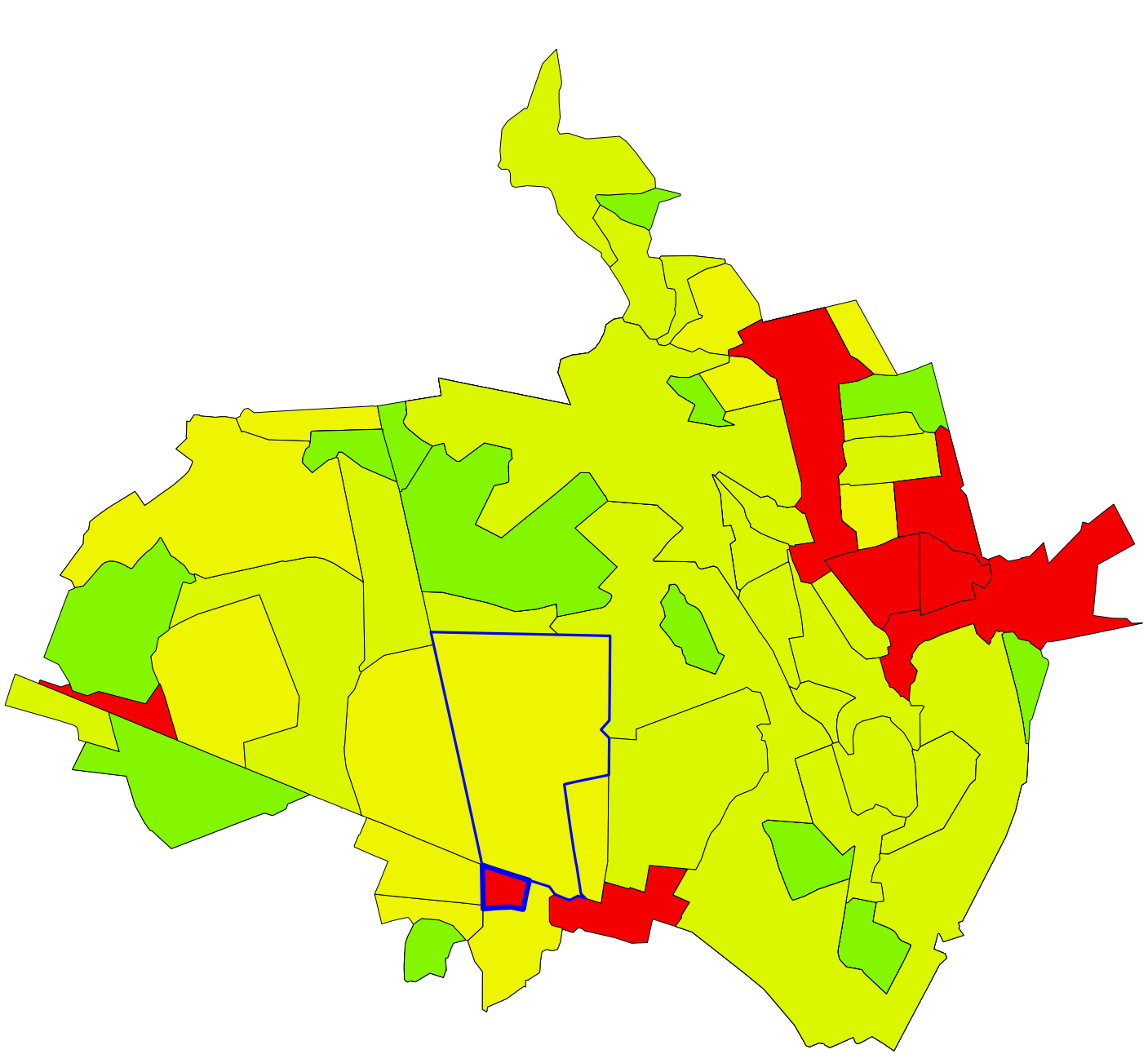


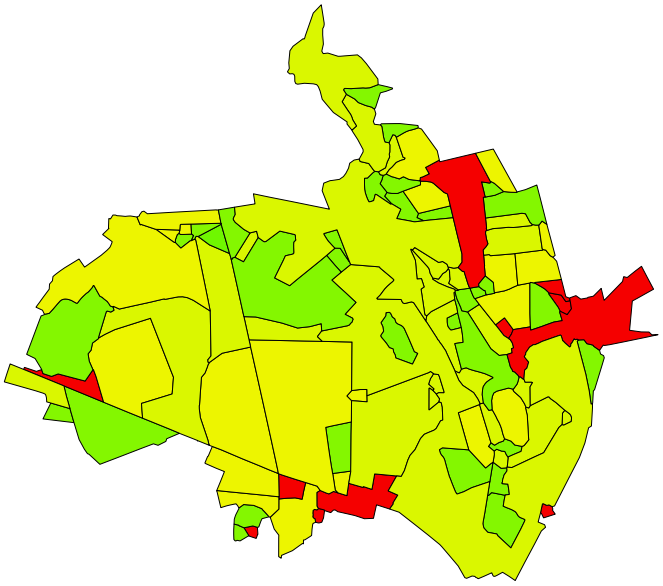


source: 92 patches

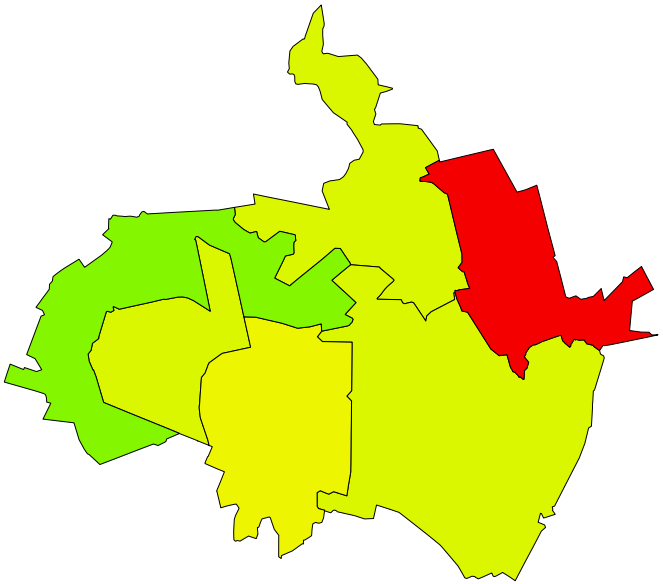


target: 6 patches

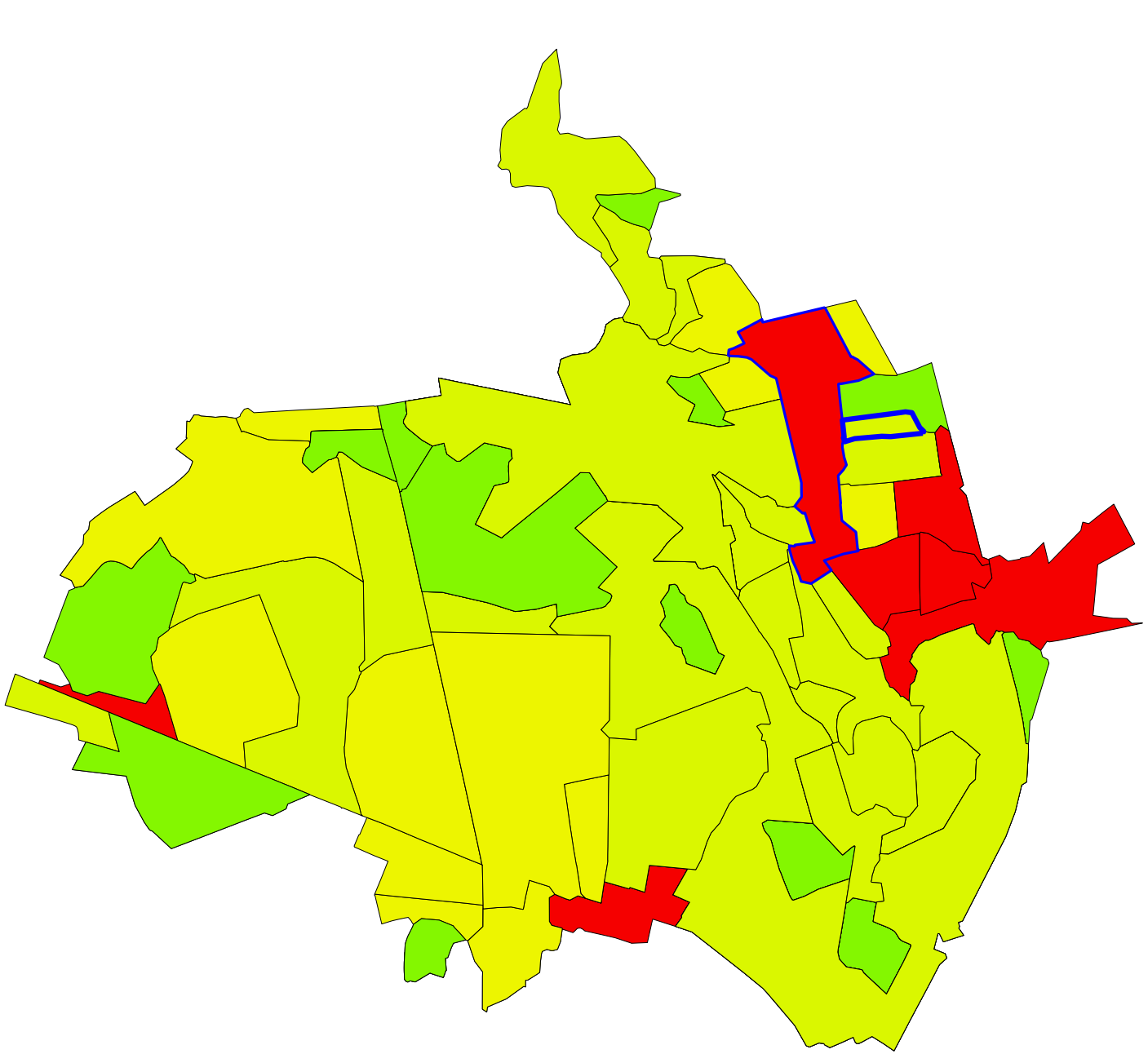


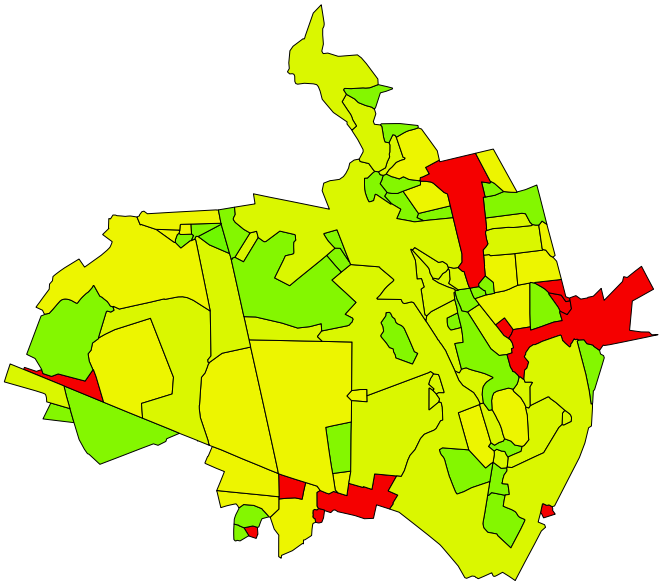


source: 92 patches

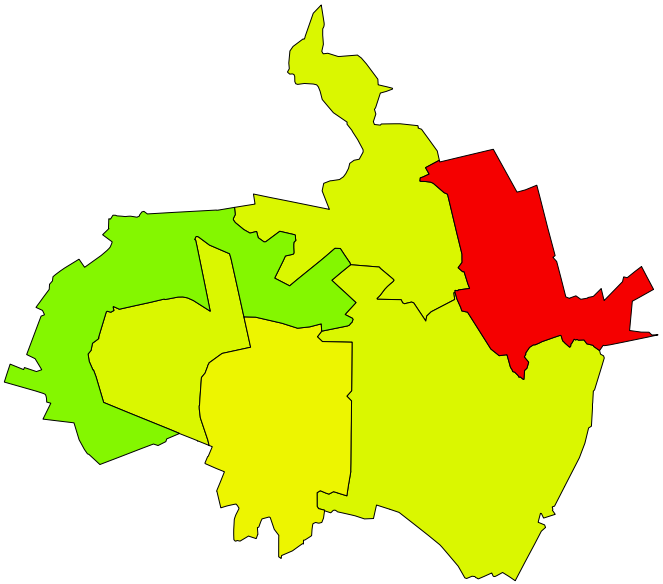


target: 6 patches

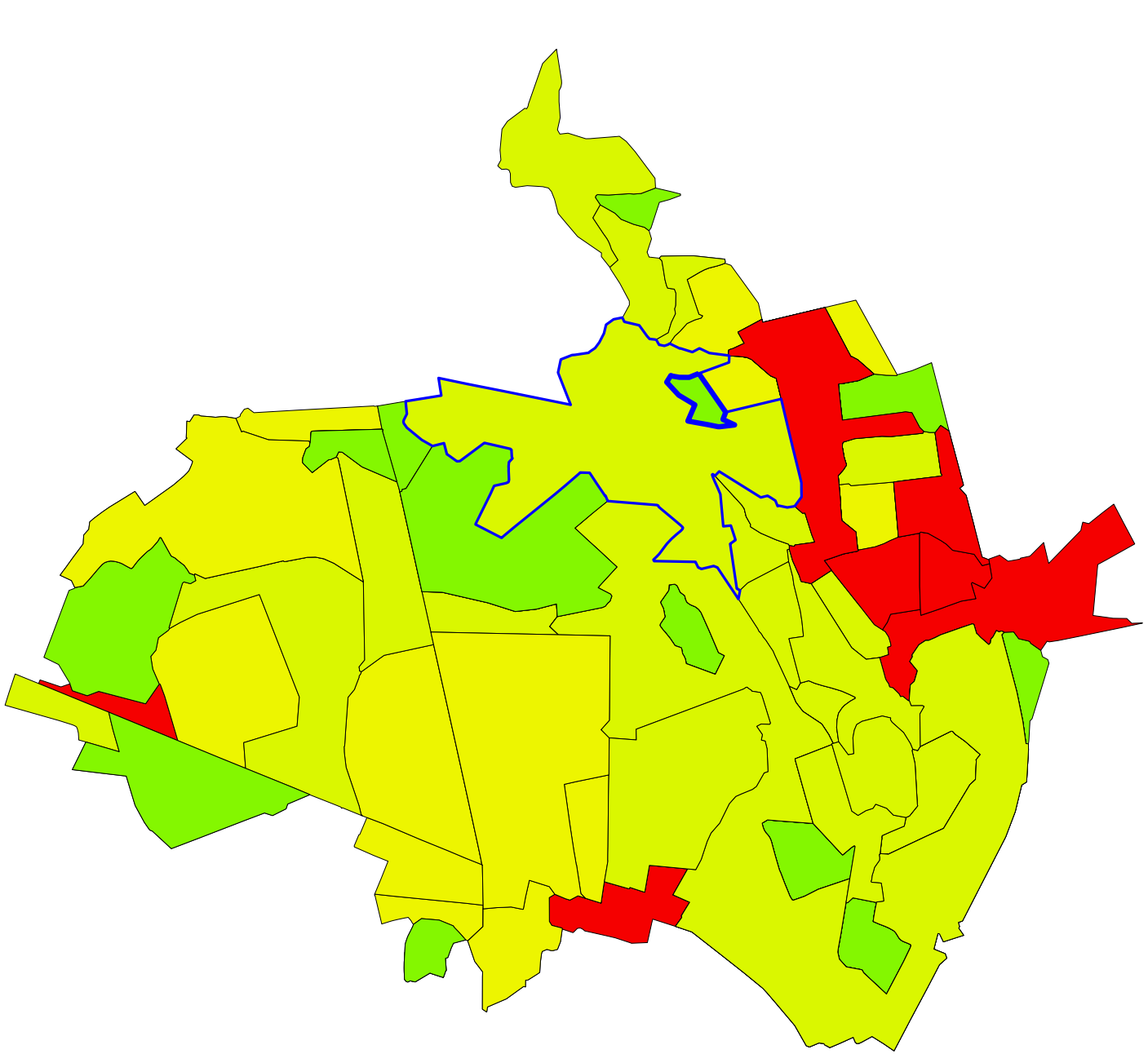


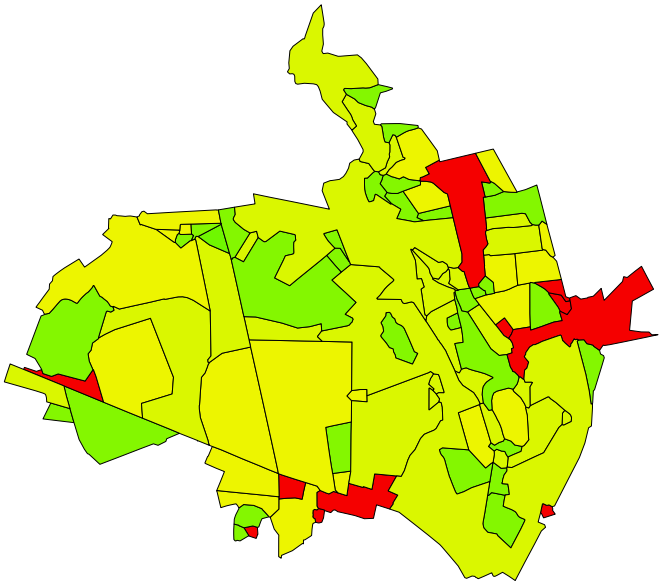


source: 92 patches

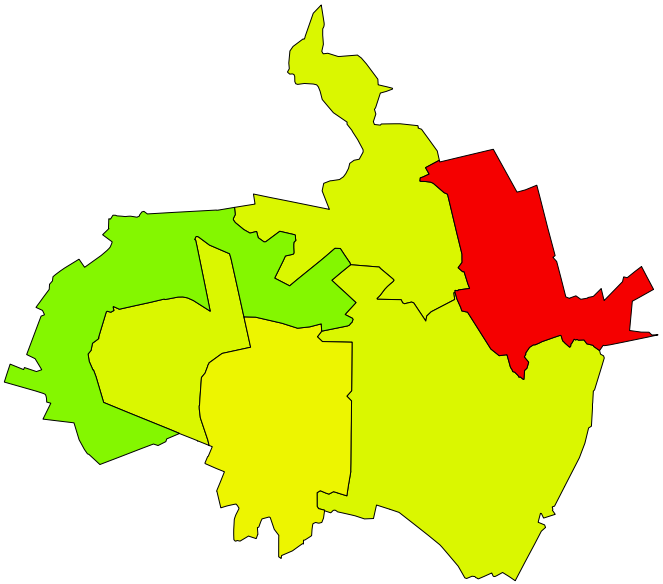


target: 6 patches

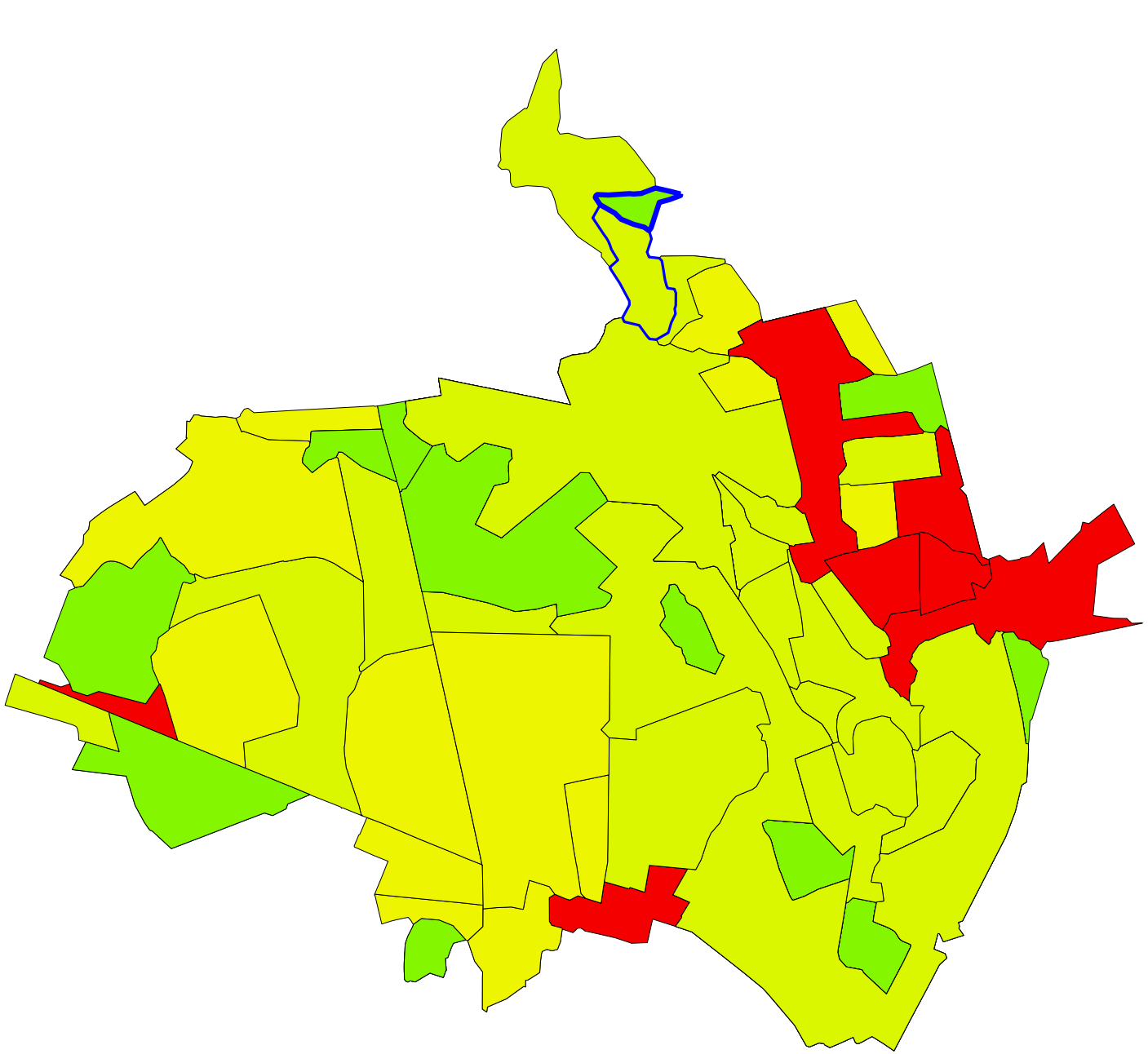


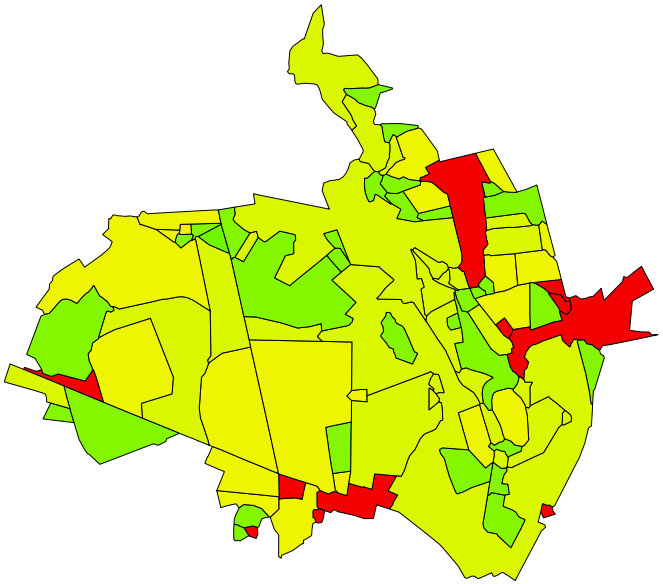


source: 92 patches

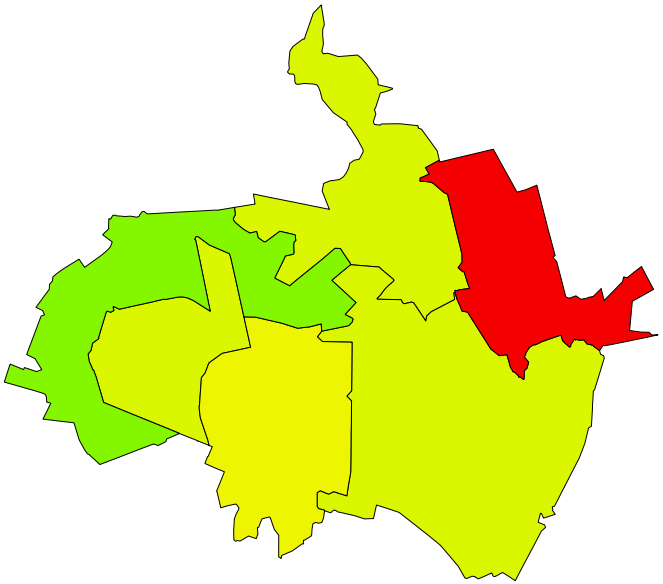


target: 6 patches

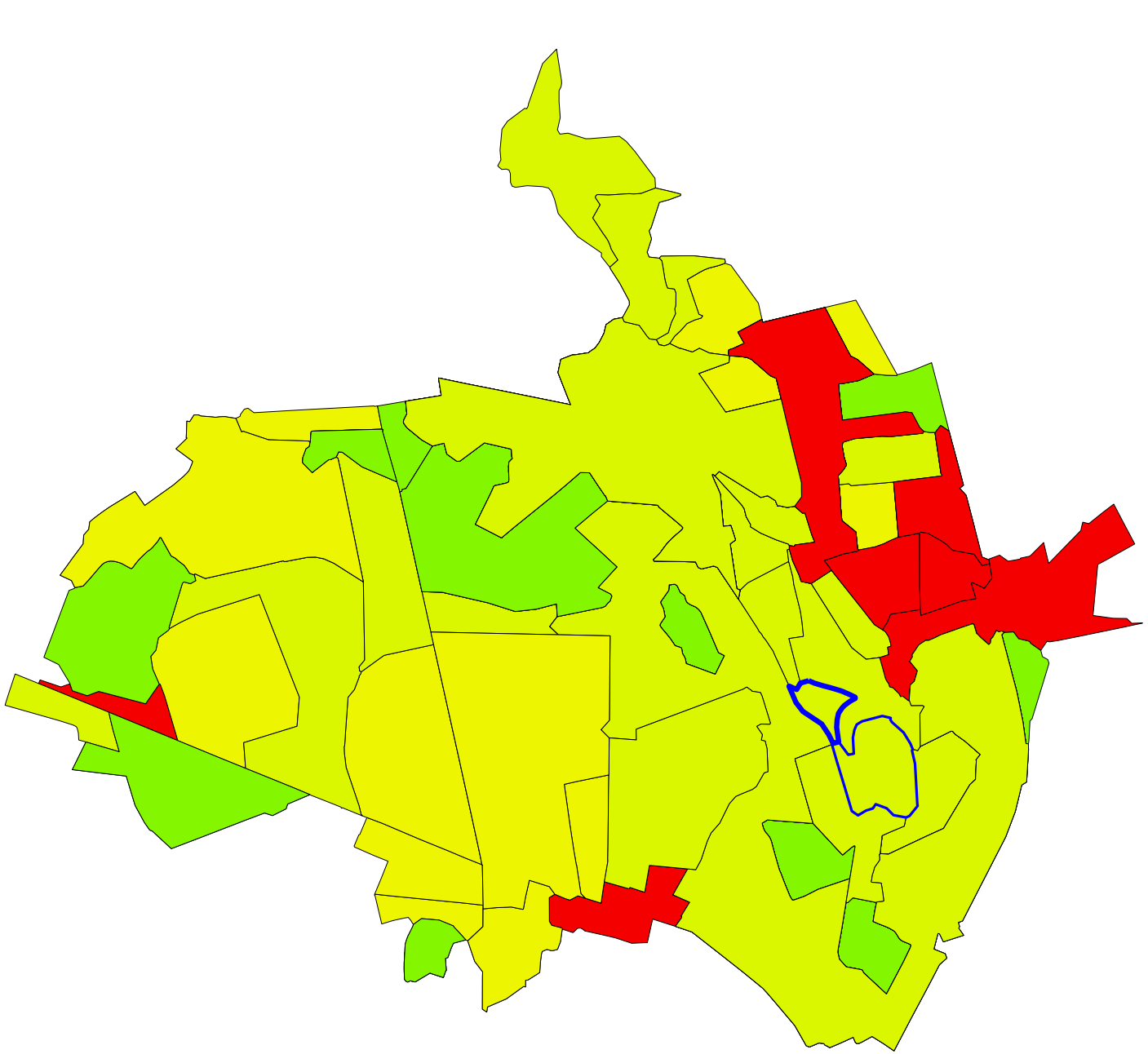


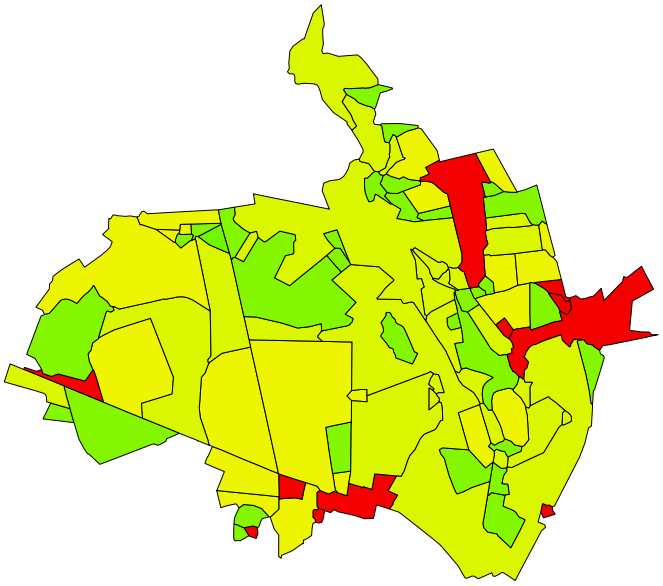


source: 92 patches

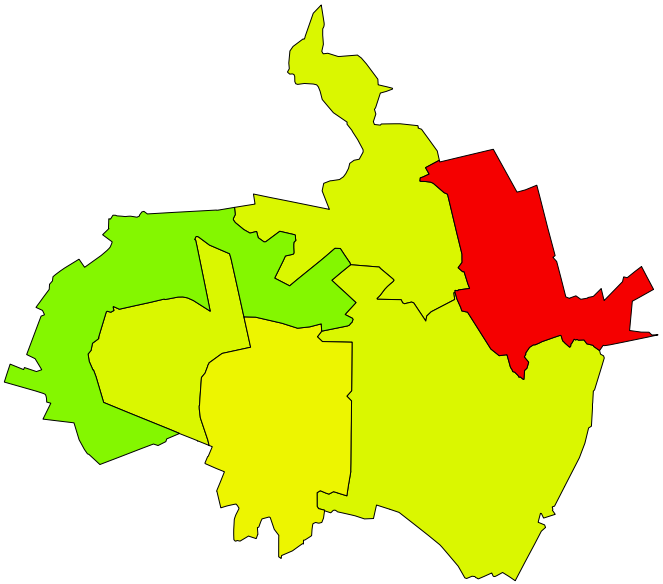


target: 6 patches

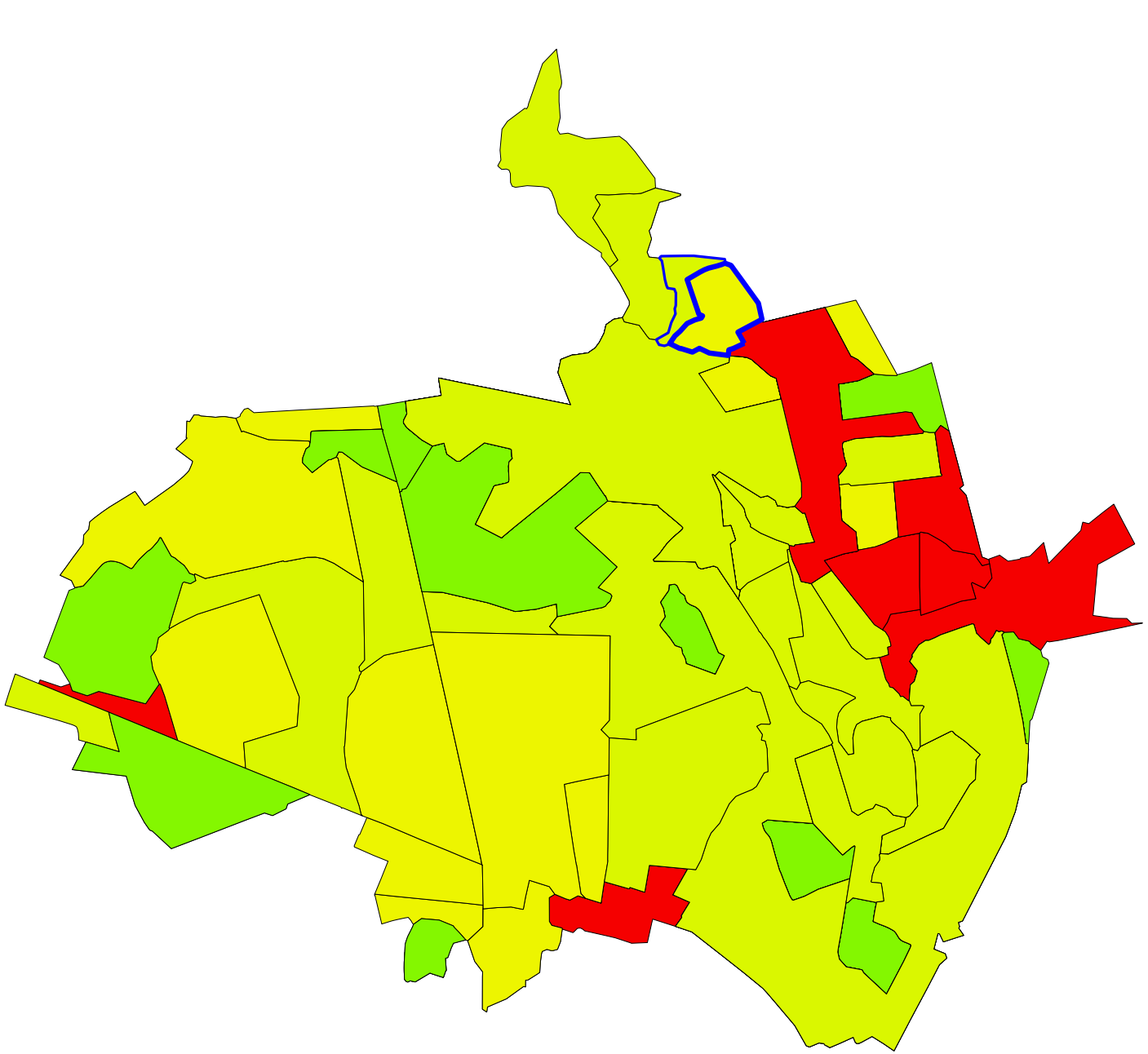


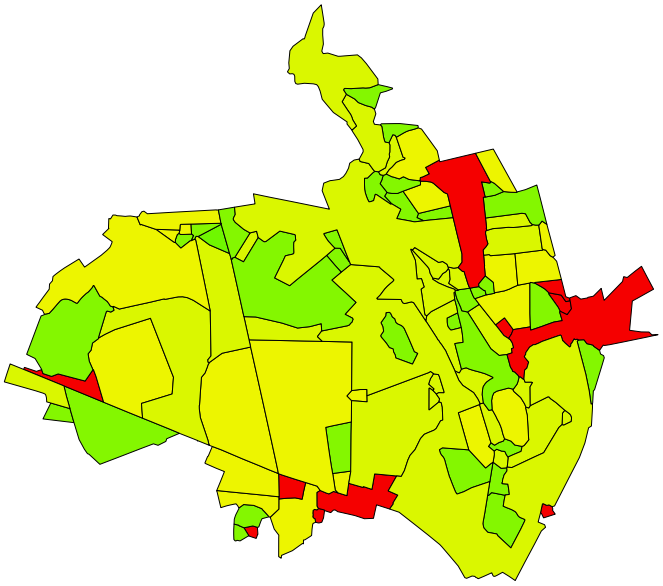


source: 92 patches

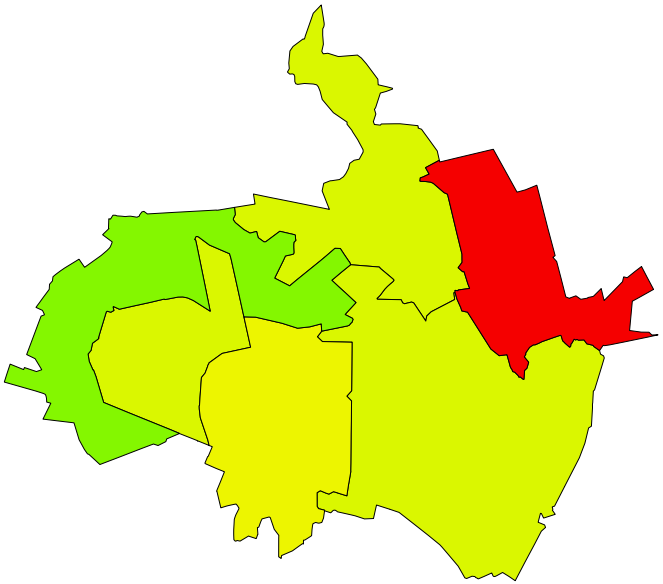


target: 6 patches

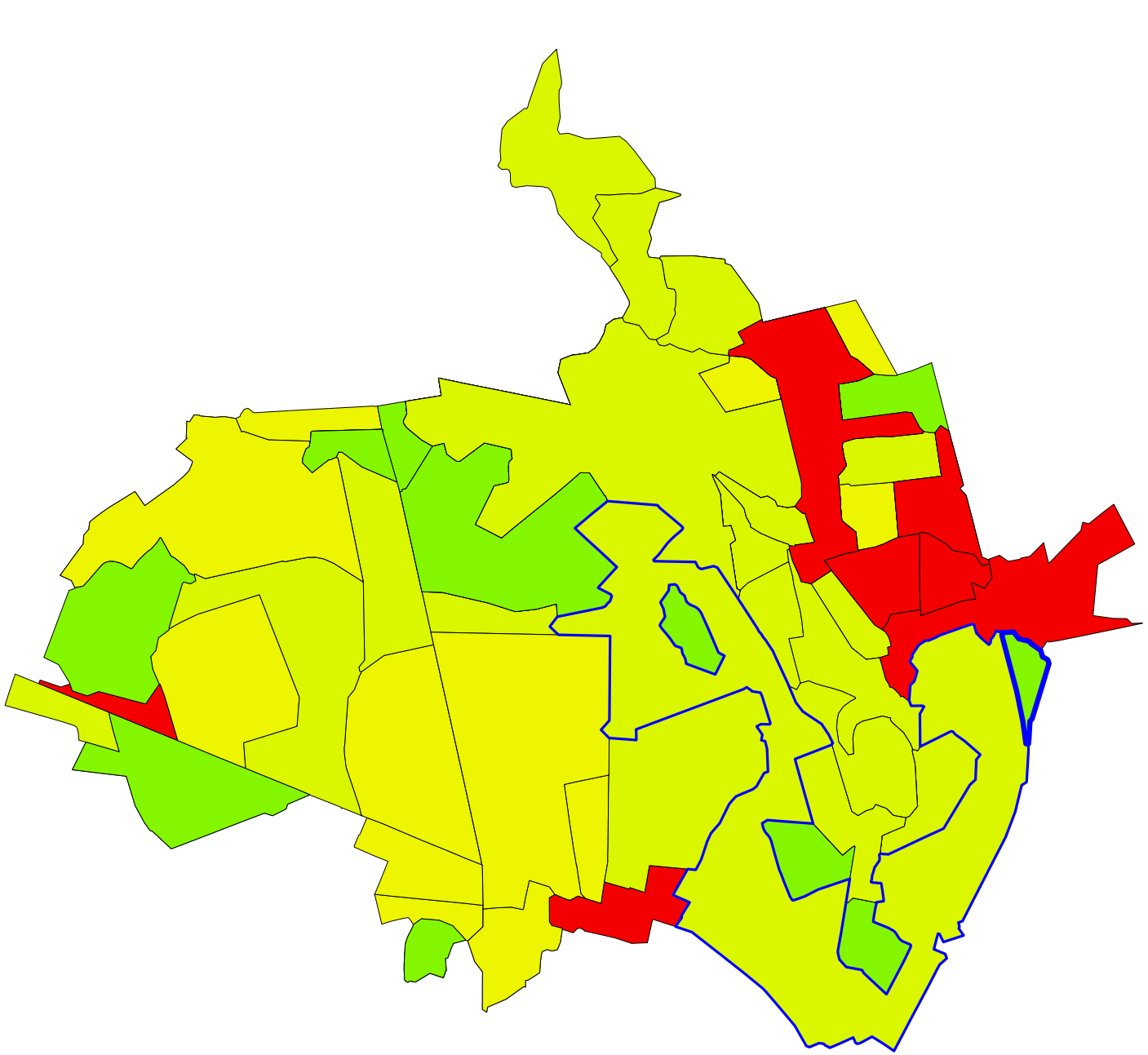


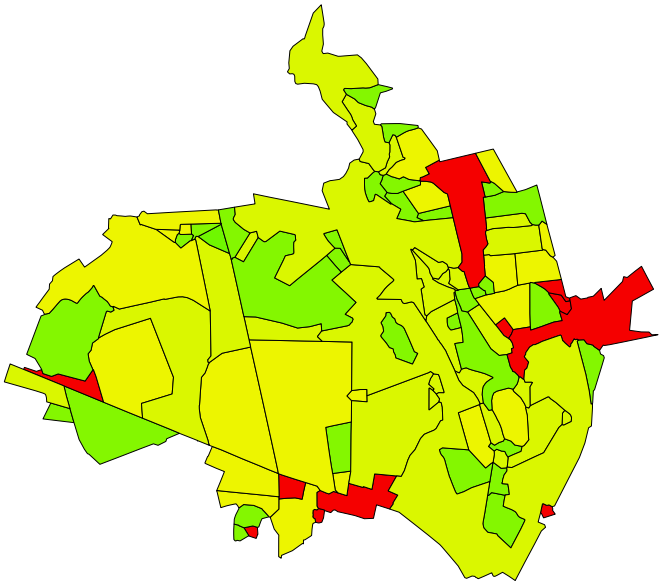


source: 92 patches

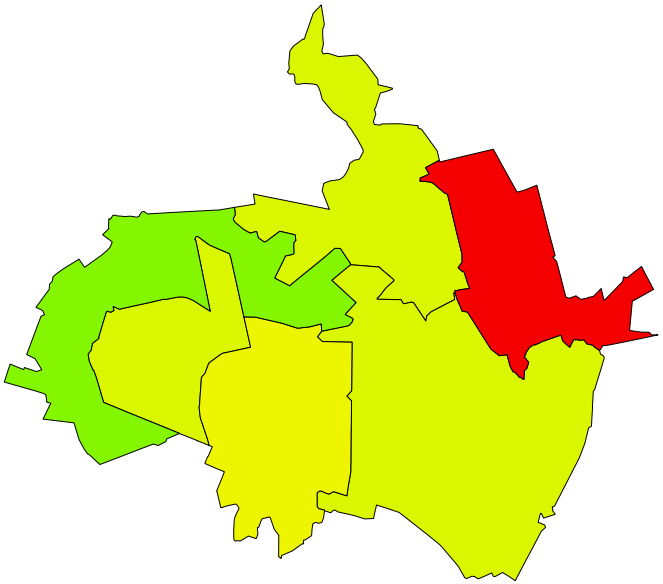


target: 6 patches

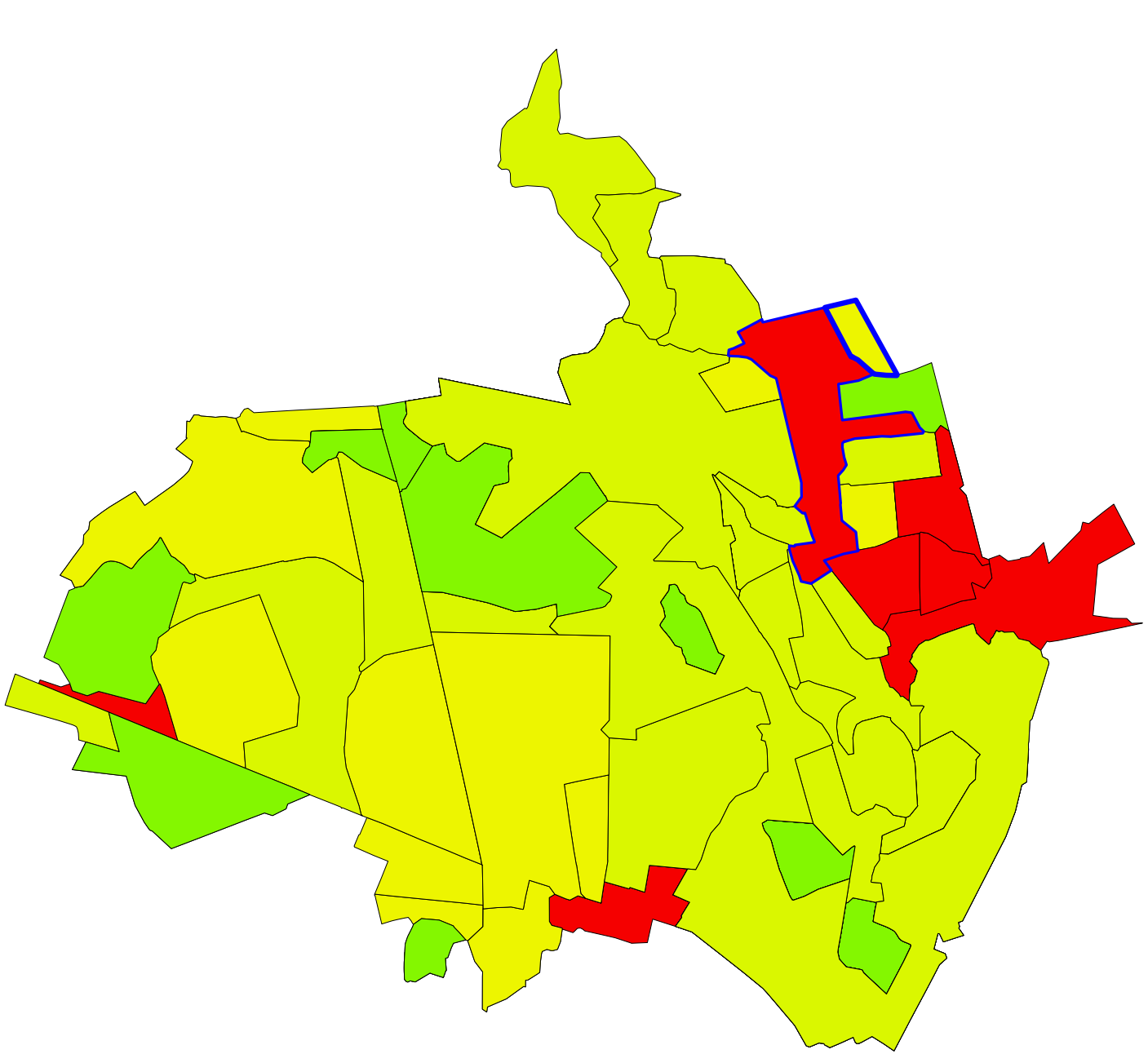


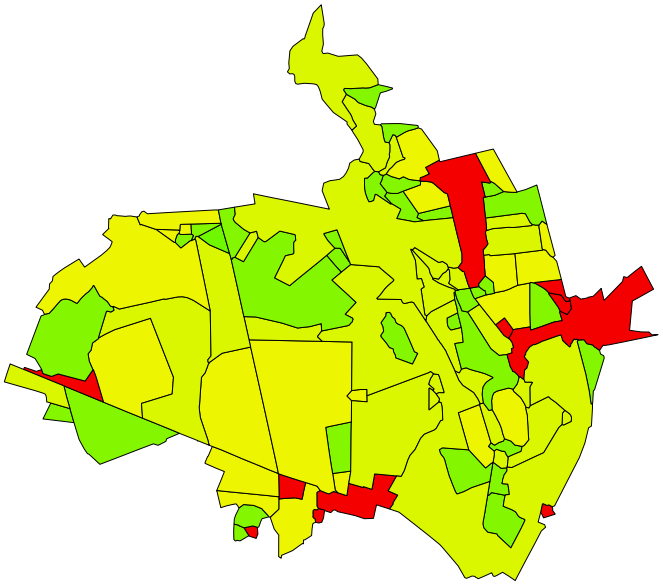


source: 92 patches

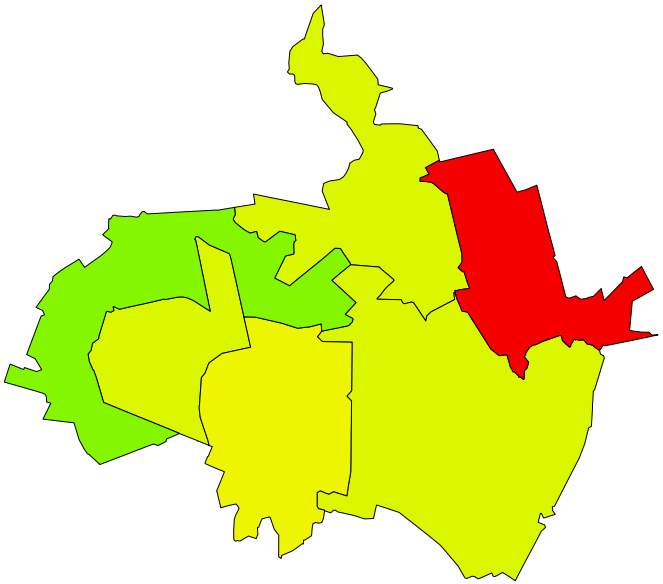


target: 6 patches

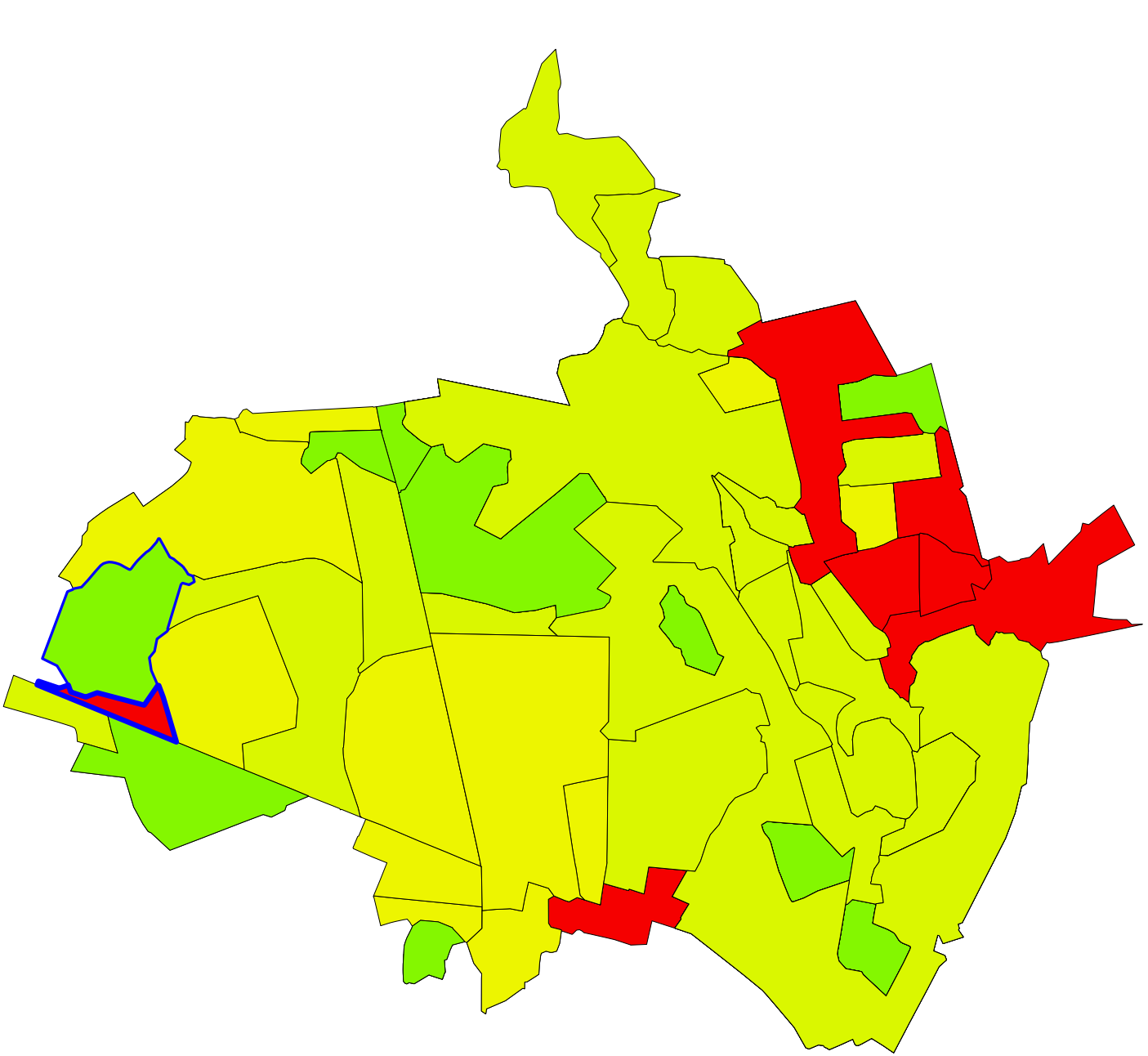


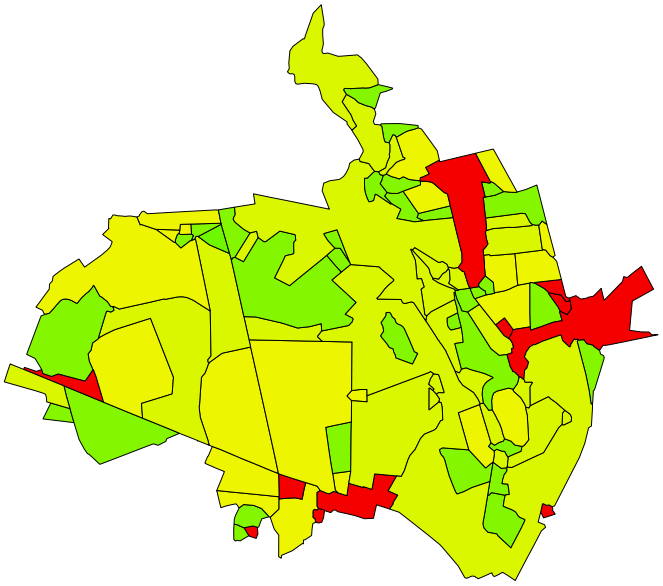


source: 92 patches

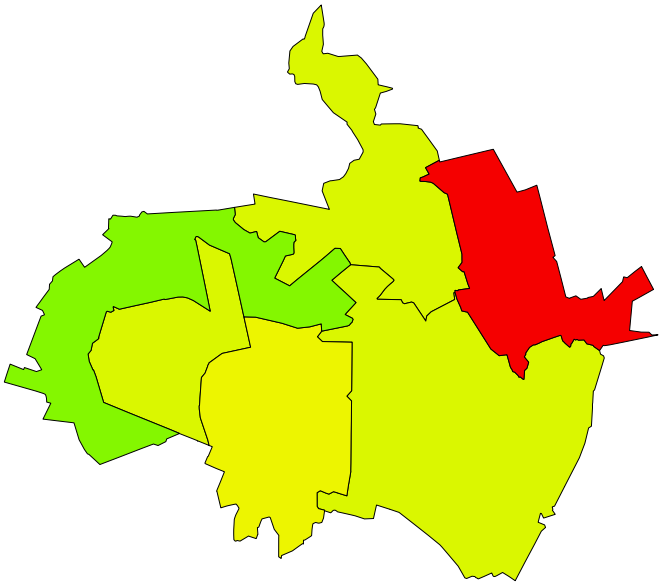


target: 6 patches

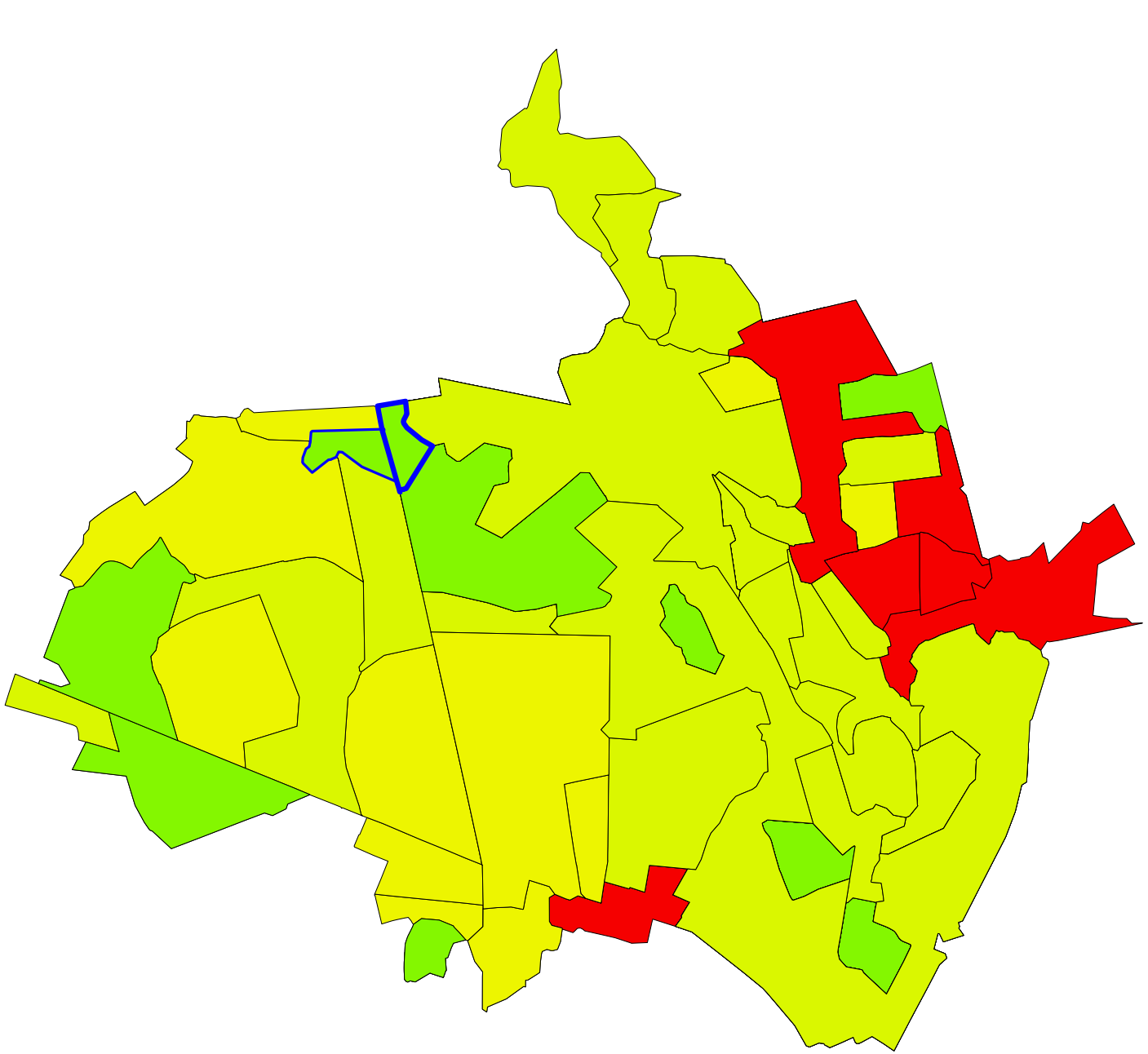


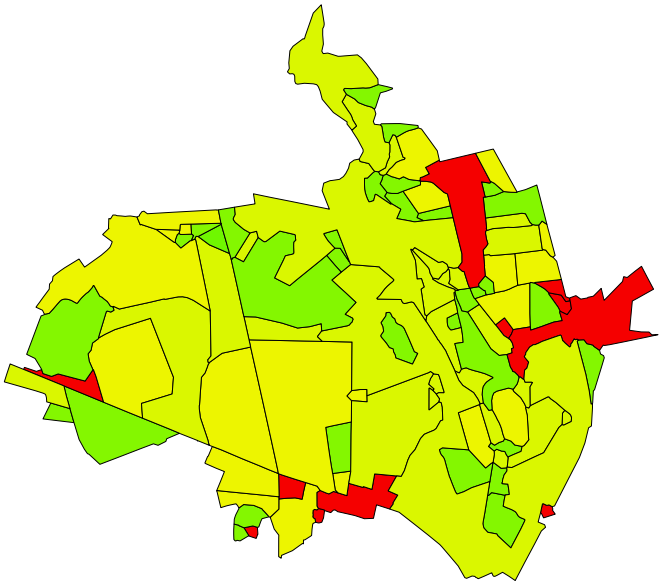


source: 92 patches

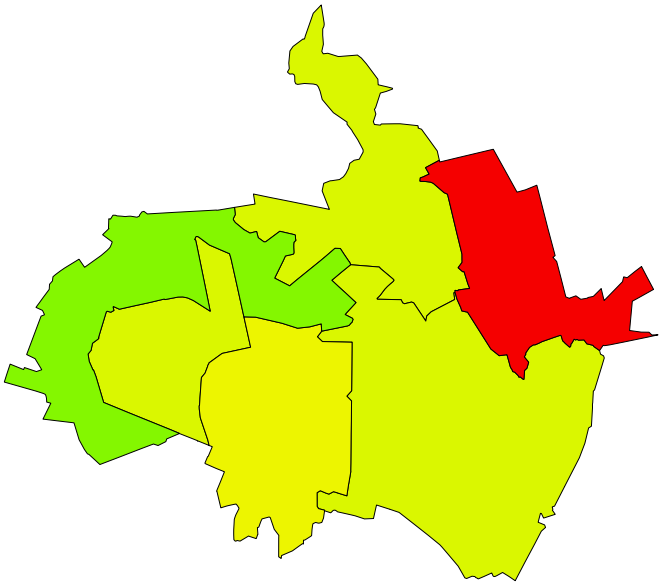


target: 6 patches

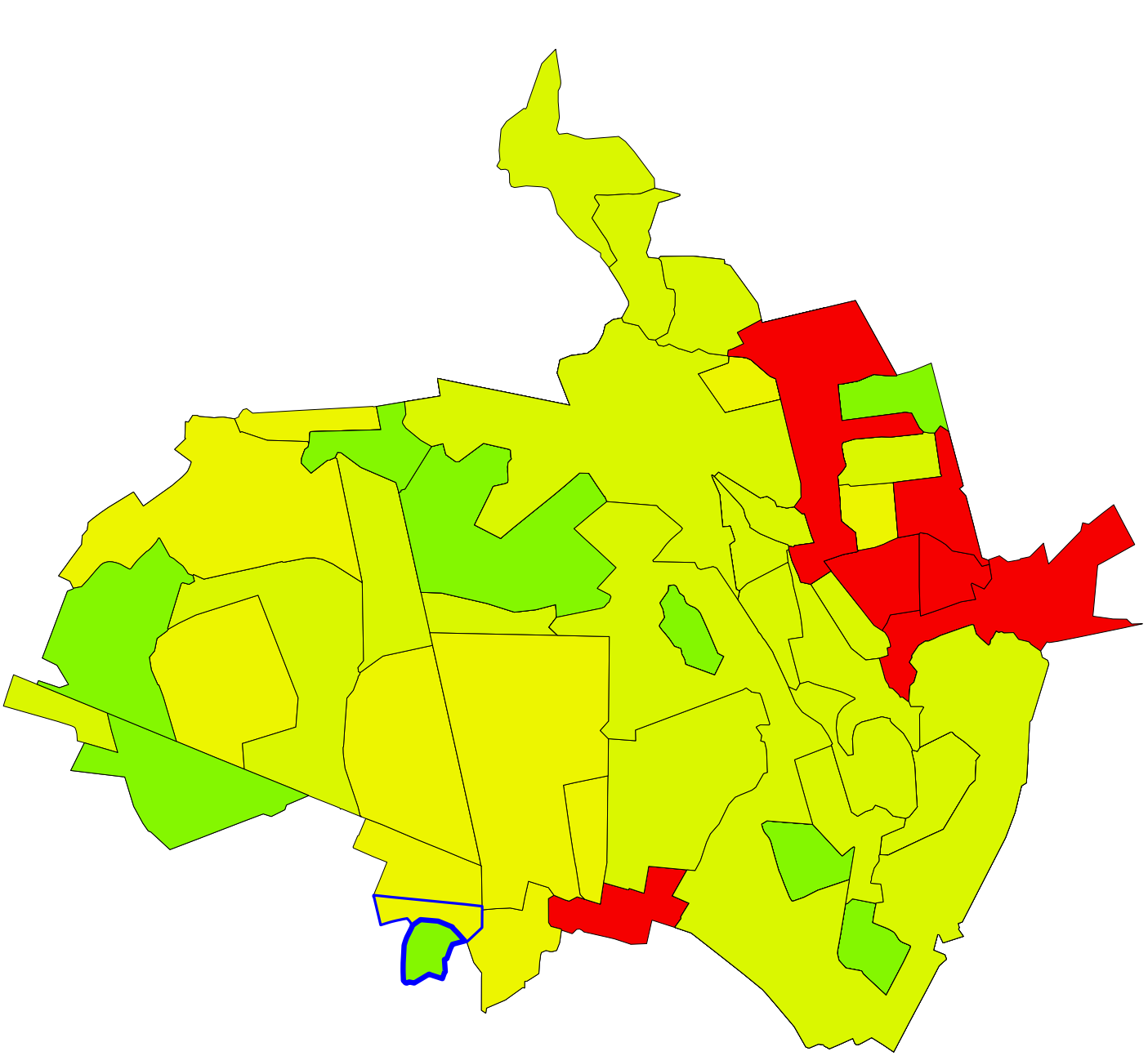


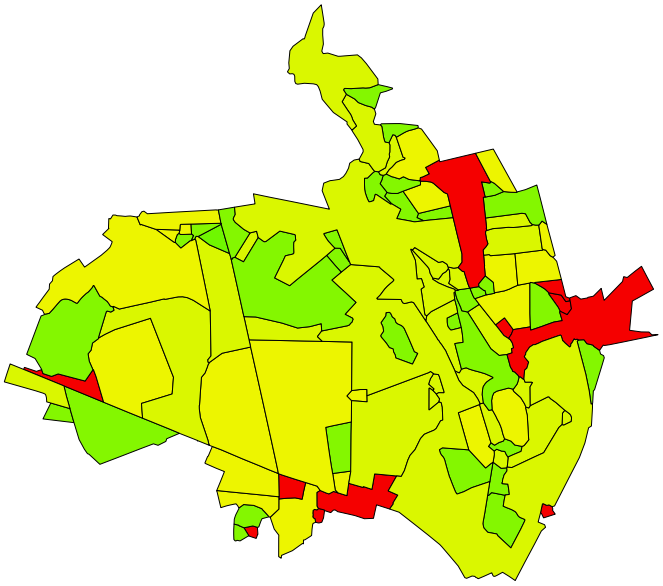


source: 92 patches

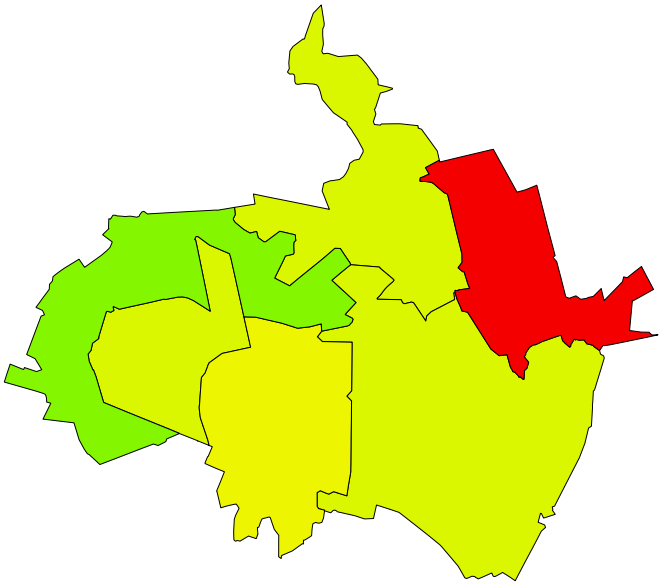


target: 6 patches

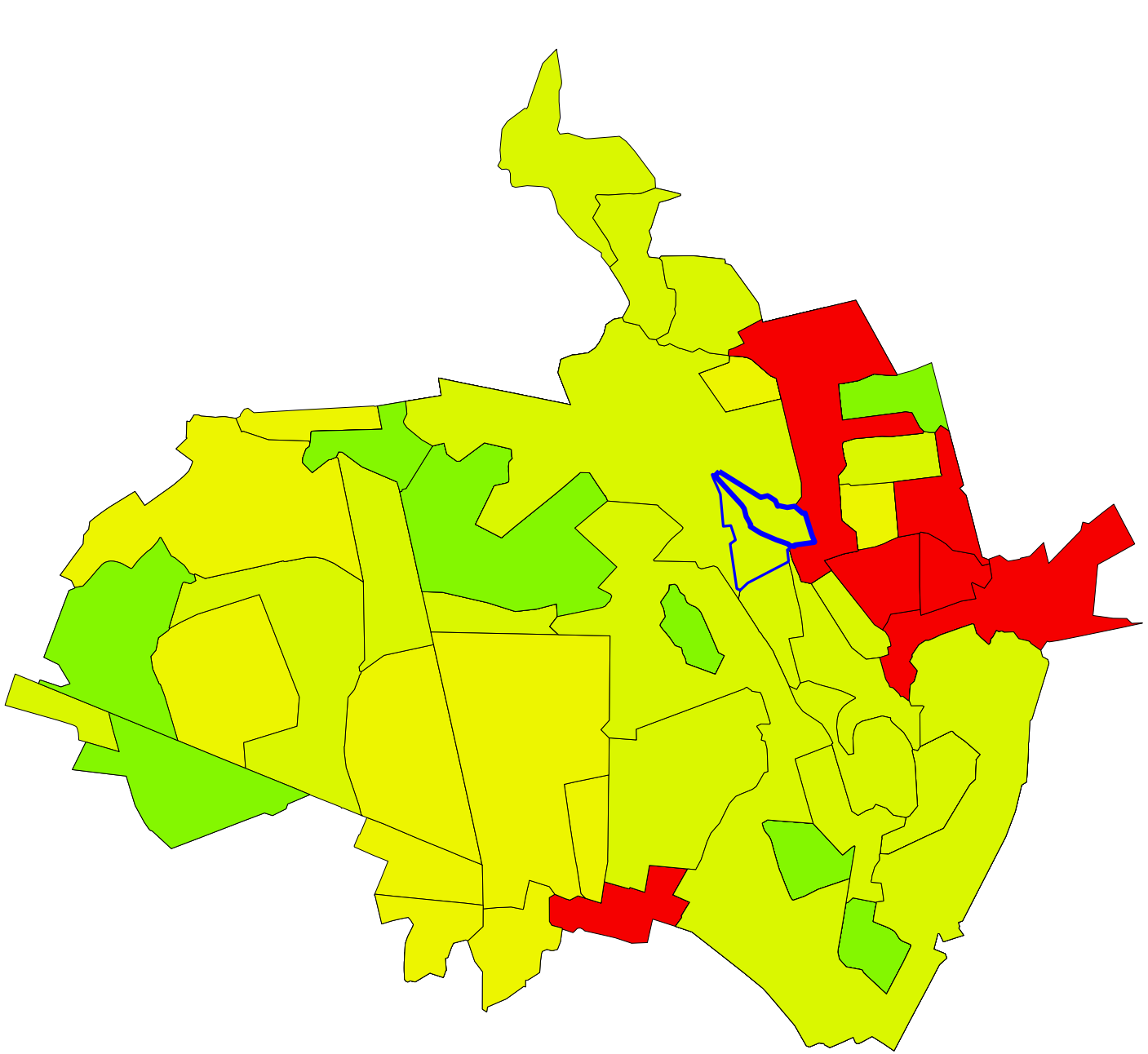


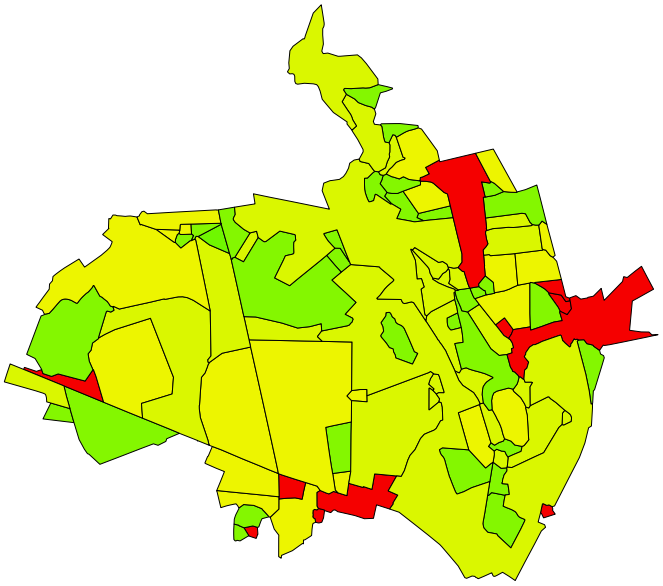


source: 92 patches

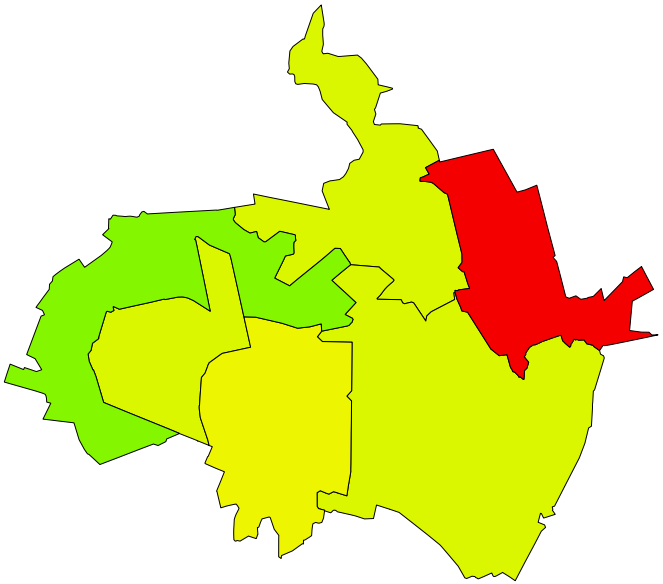


target: 6 patches

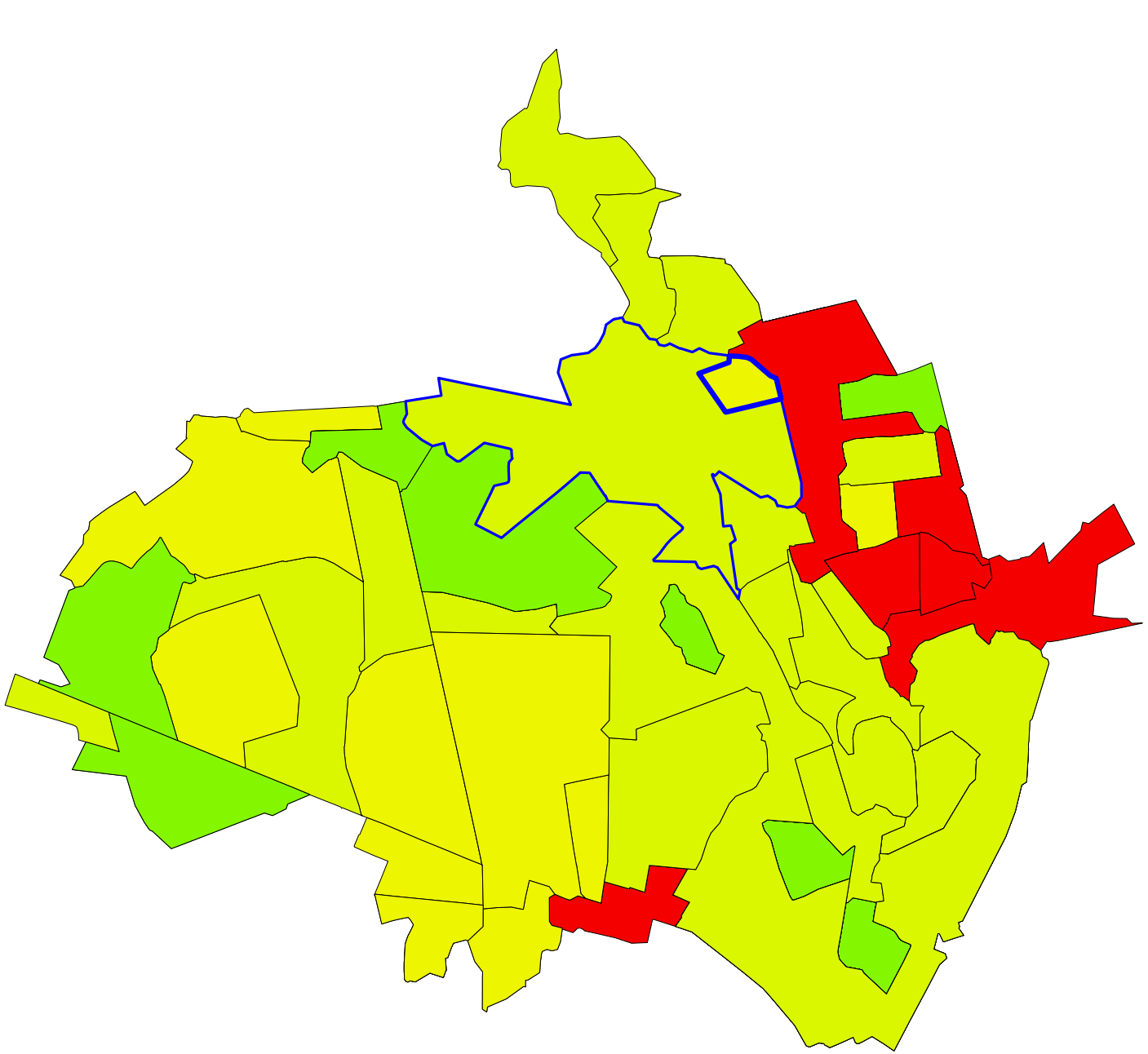


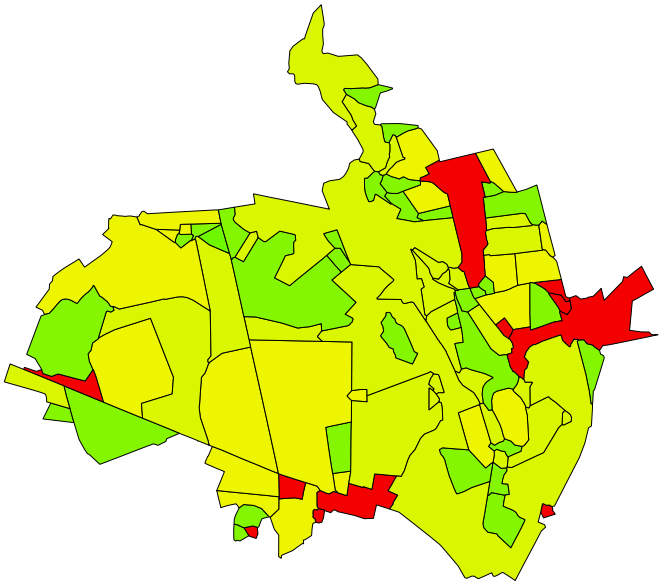


source: 92 patches

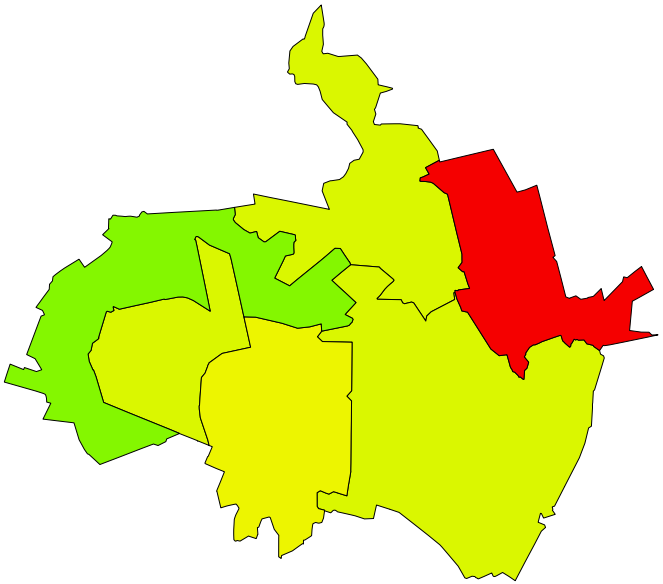


target: 6 patches

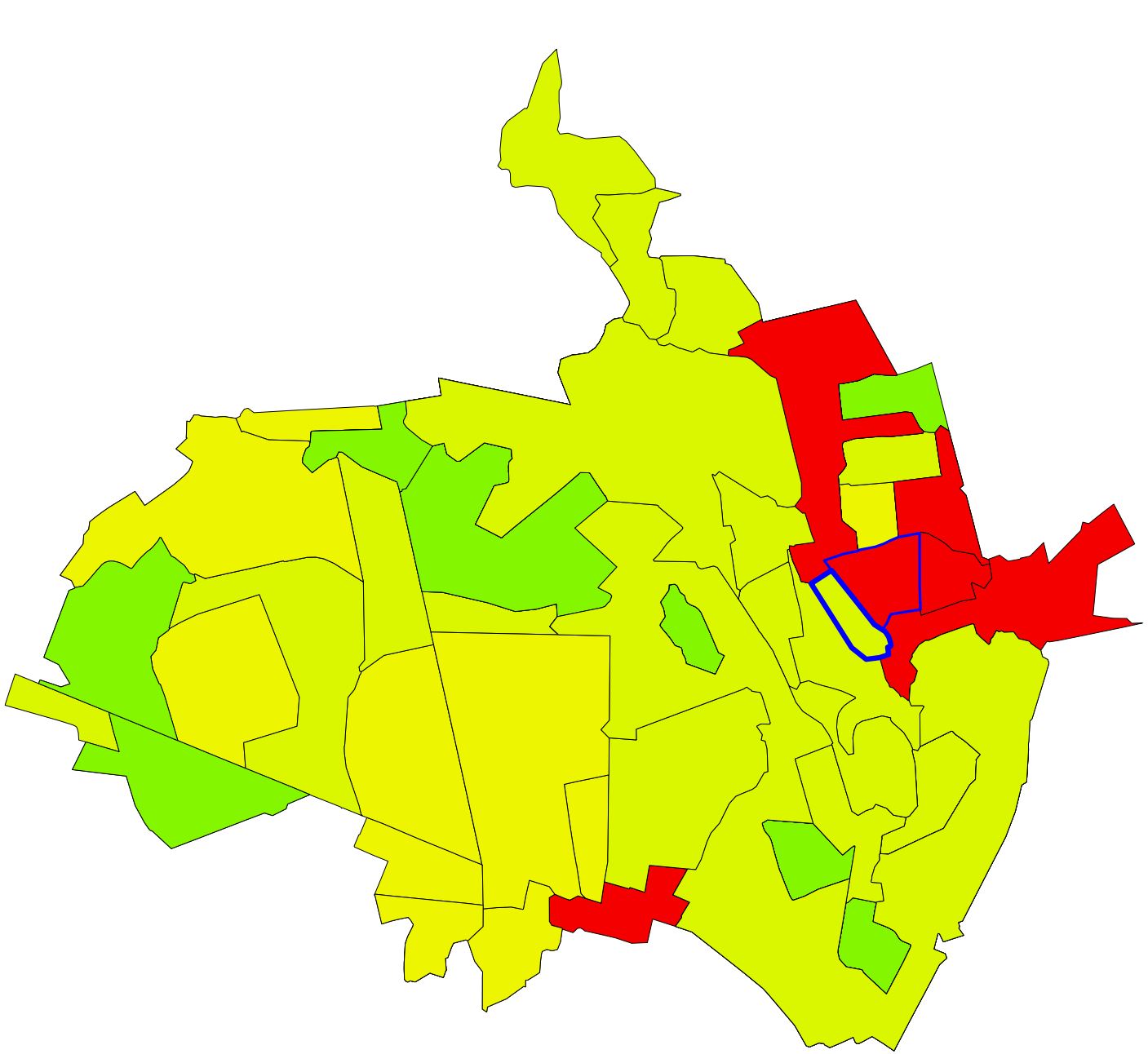


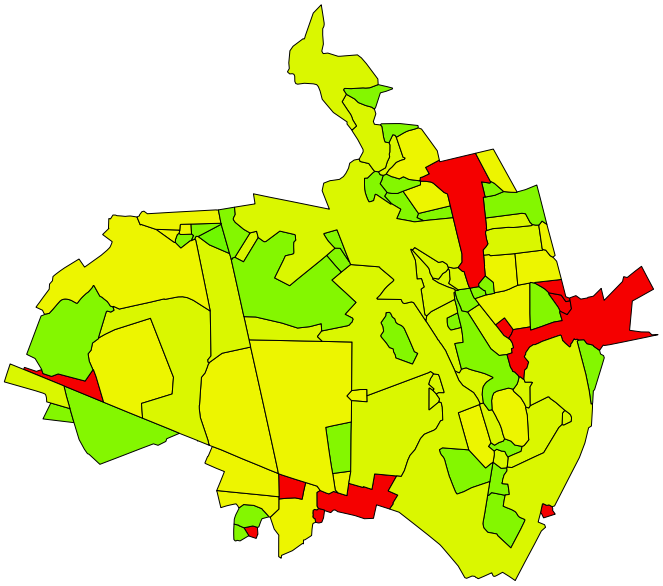


source: 92 patches

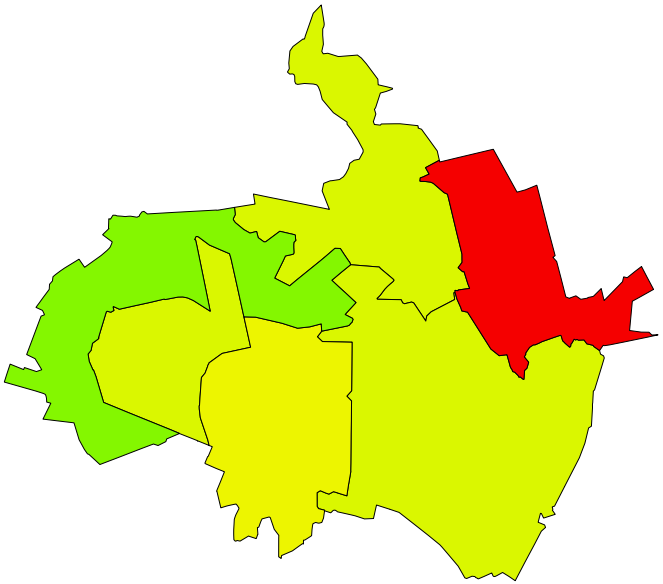


target: 6 patches

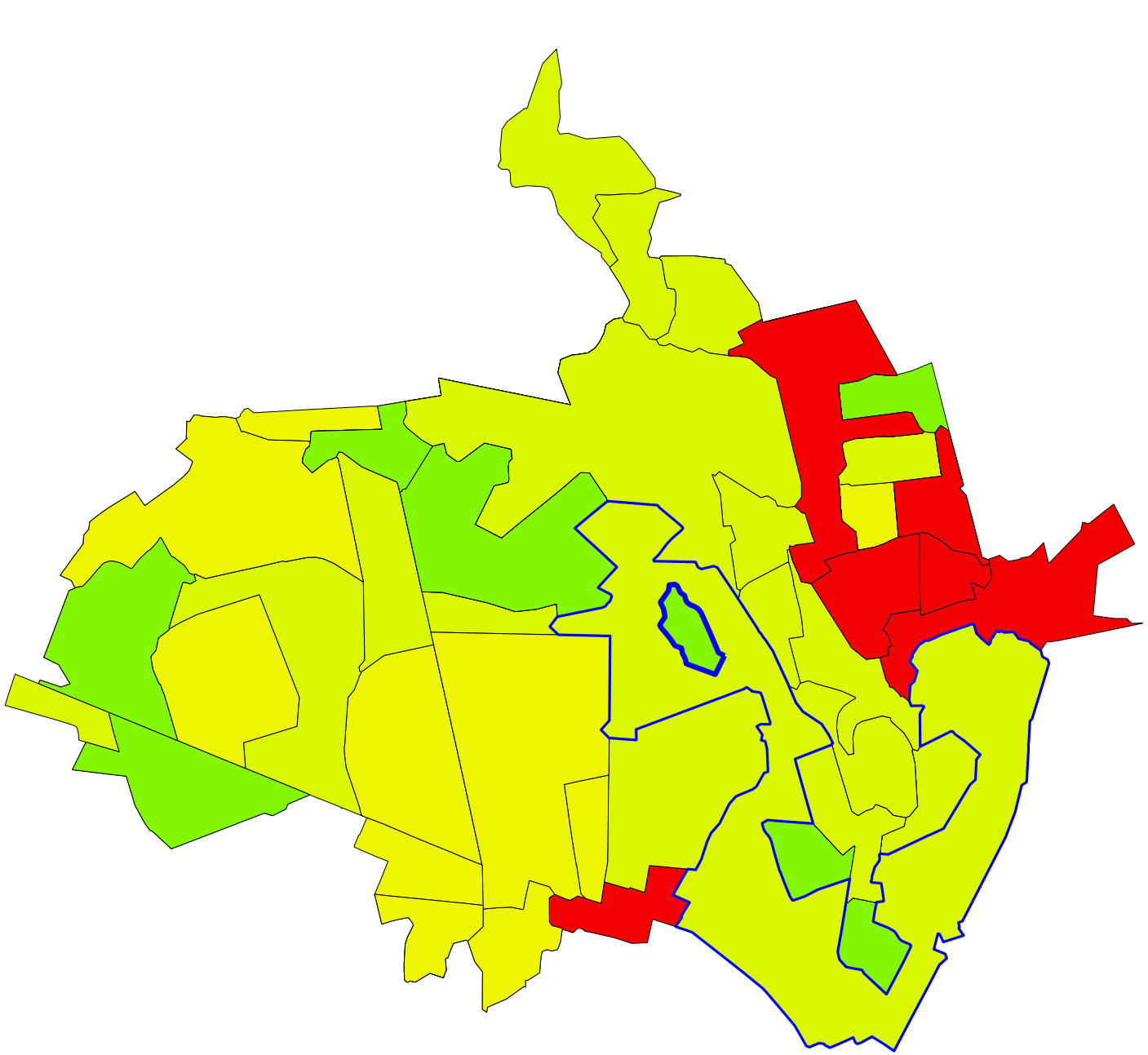


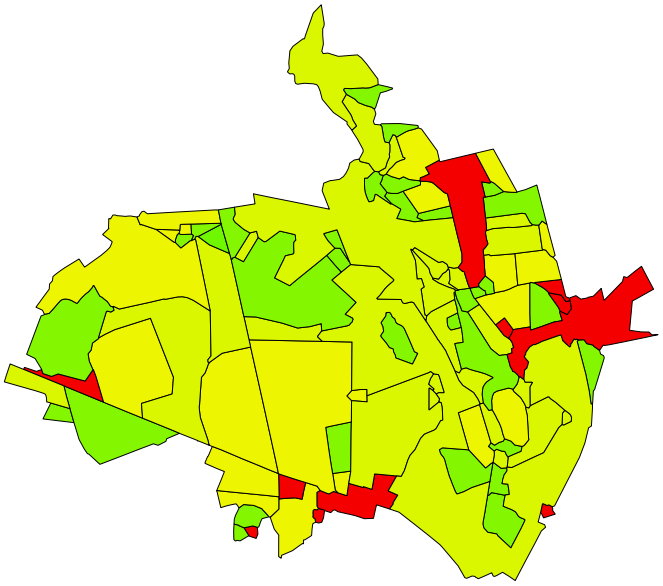


source: 92 patches

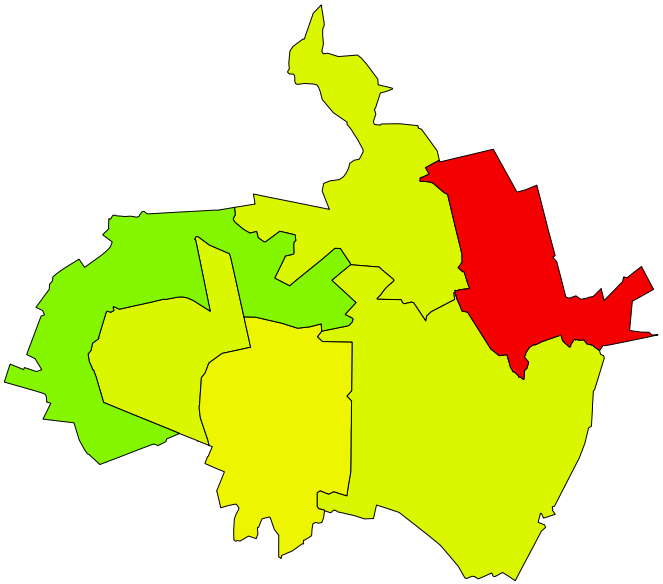


target: 6 patches

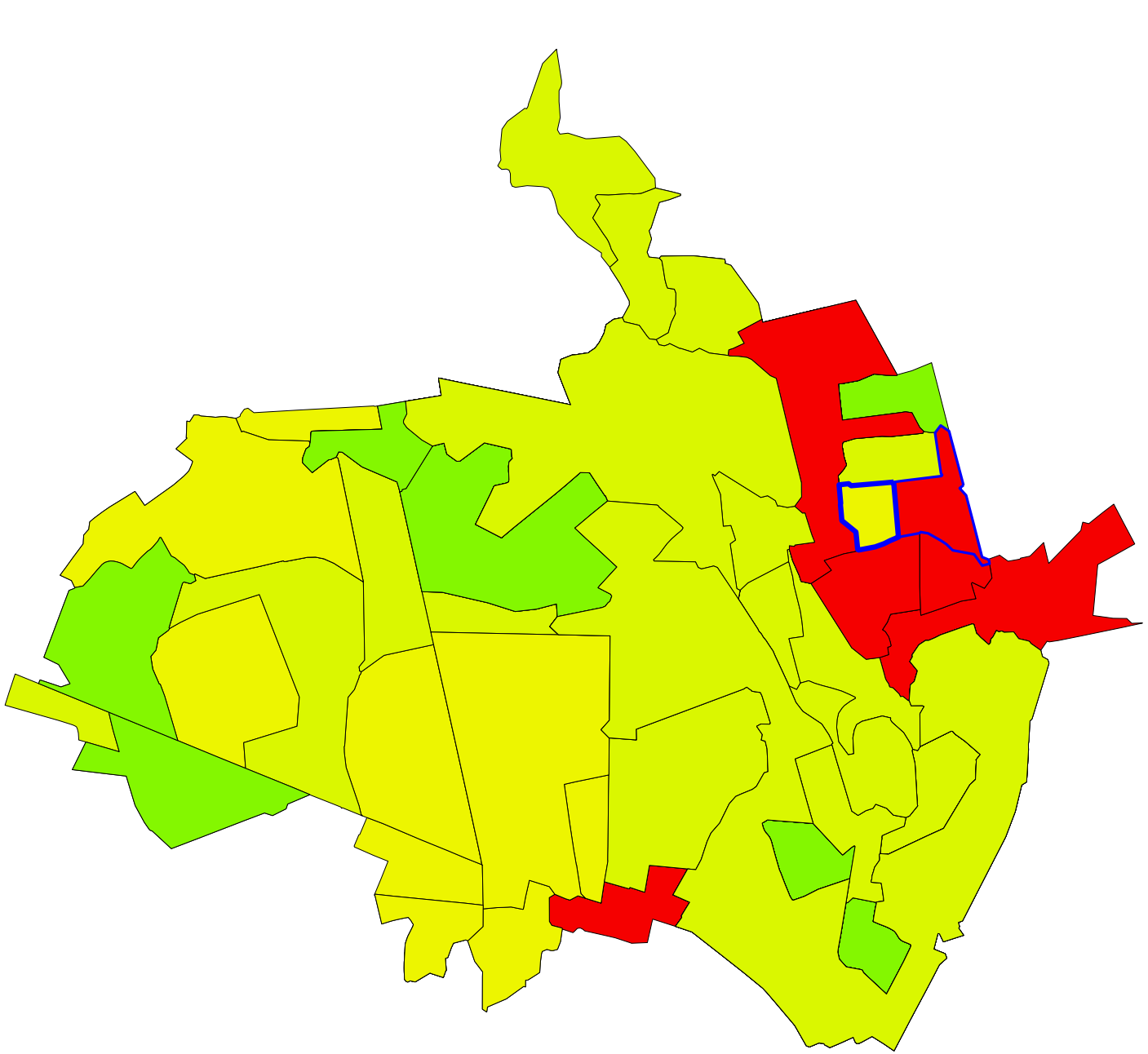


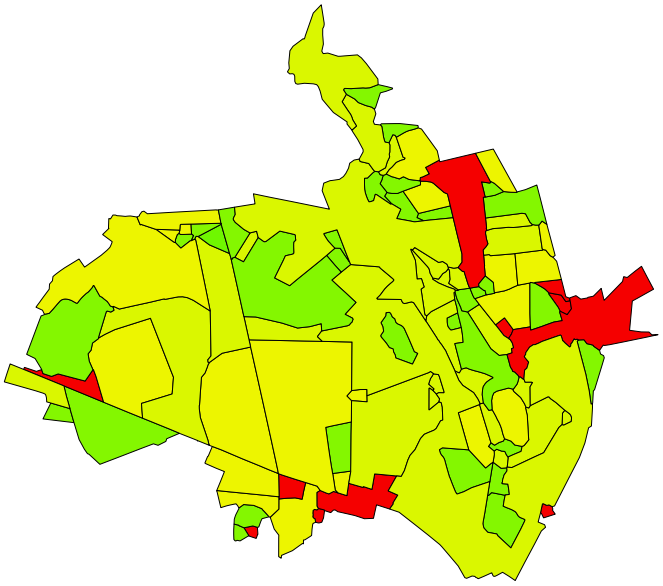


source: 92 patches

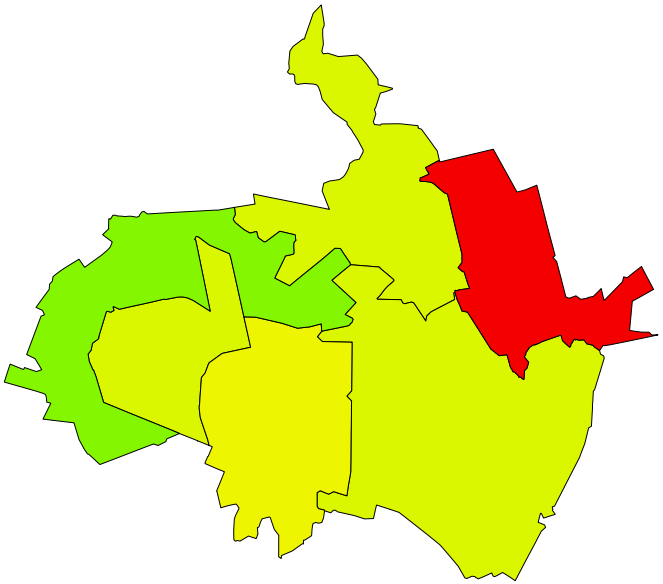


target: 6 patches

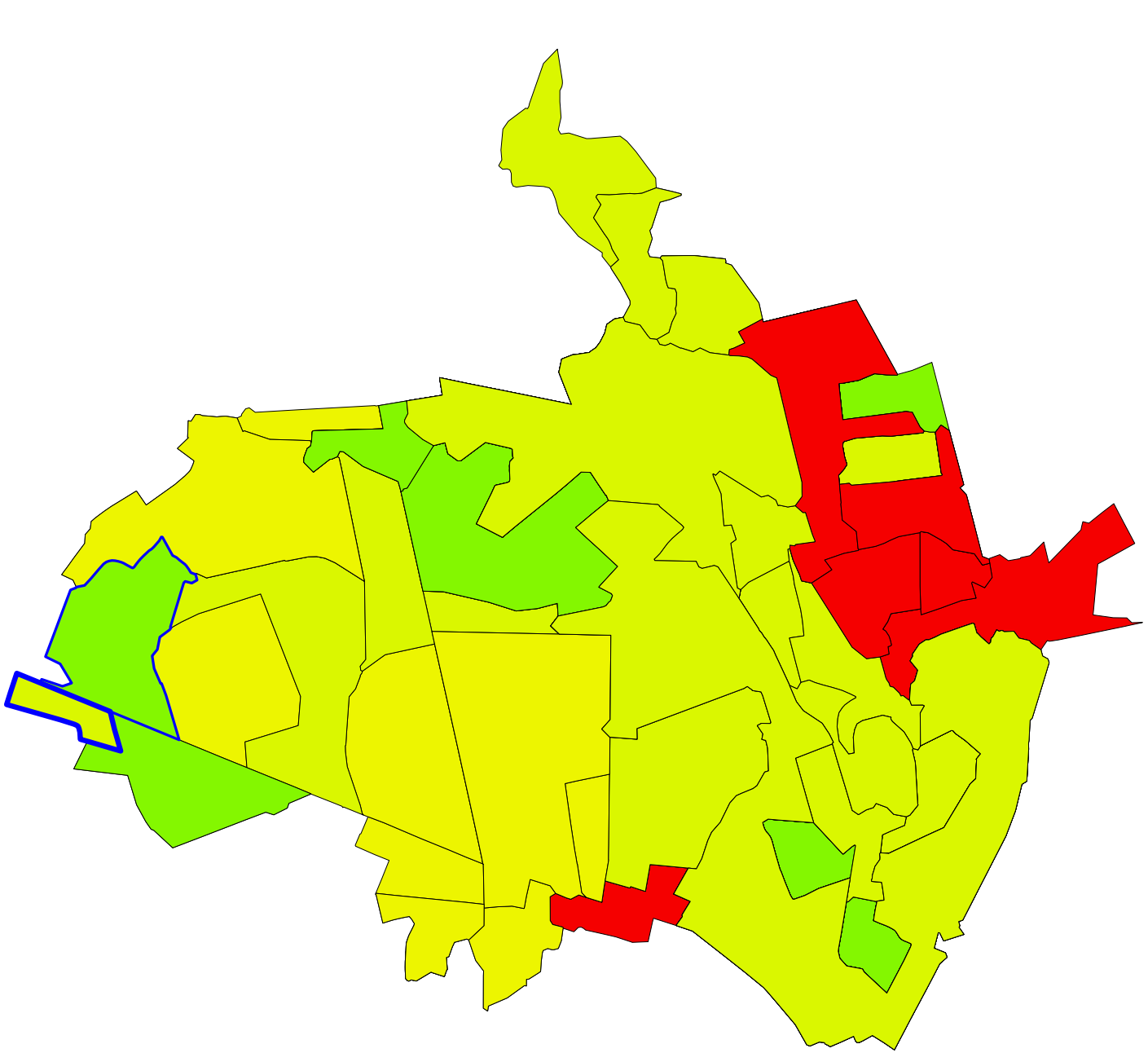


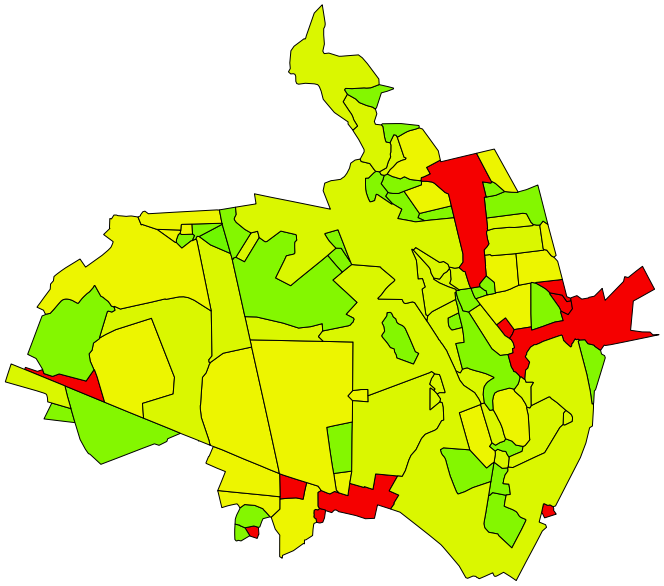


source: 92 patches

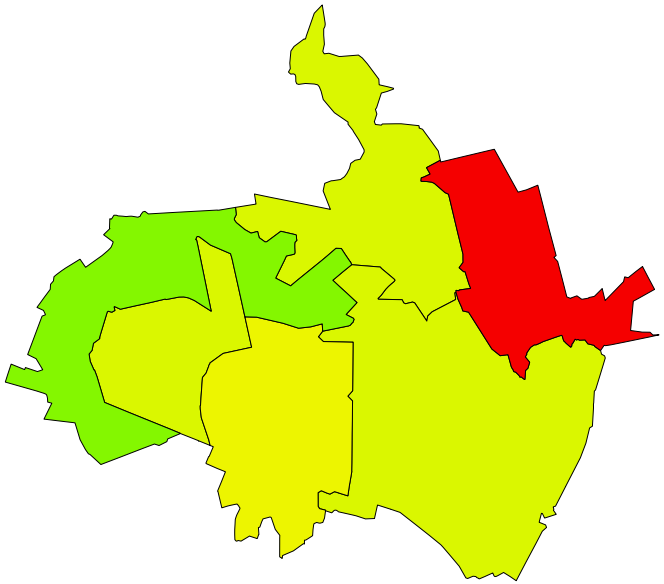


target: 6 patches

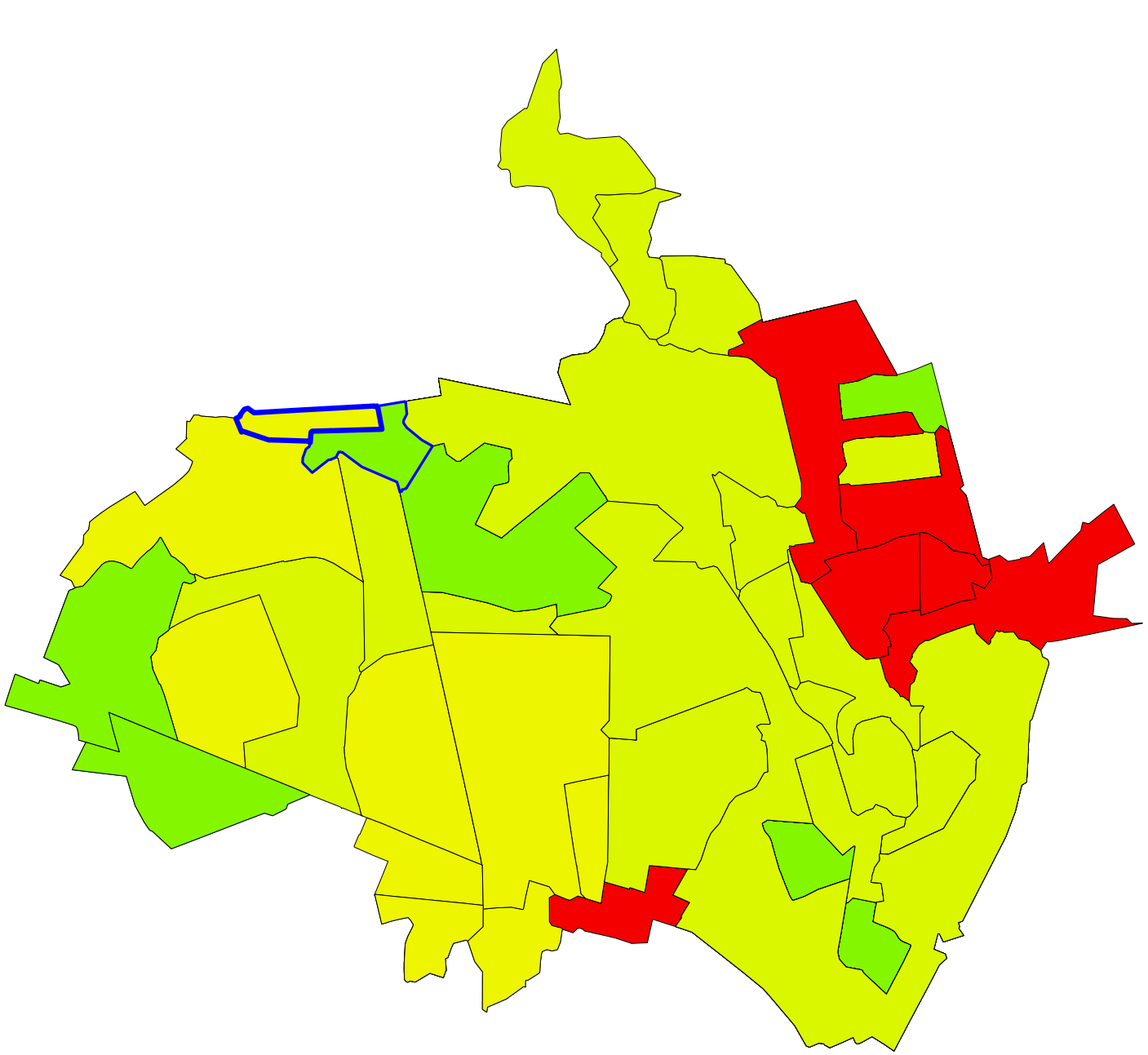


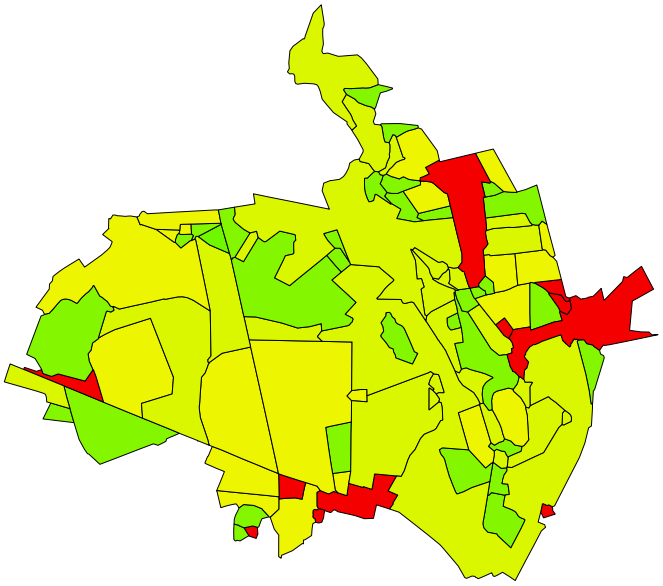


source: 92 patches

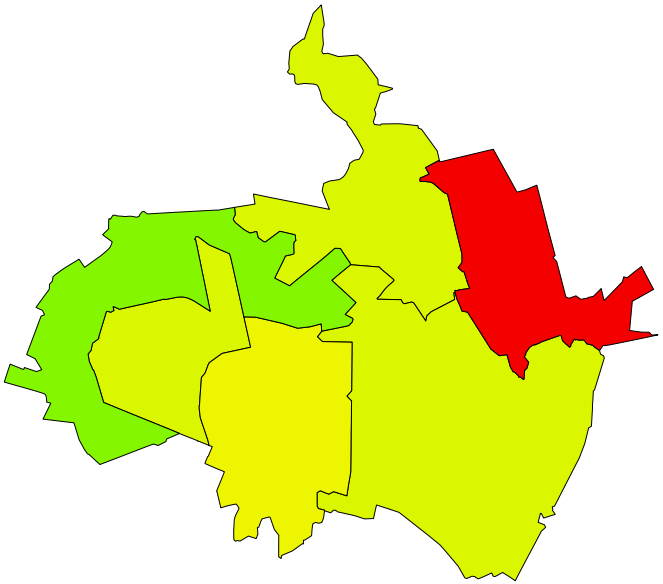


target: 6 patches

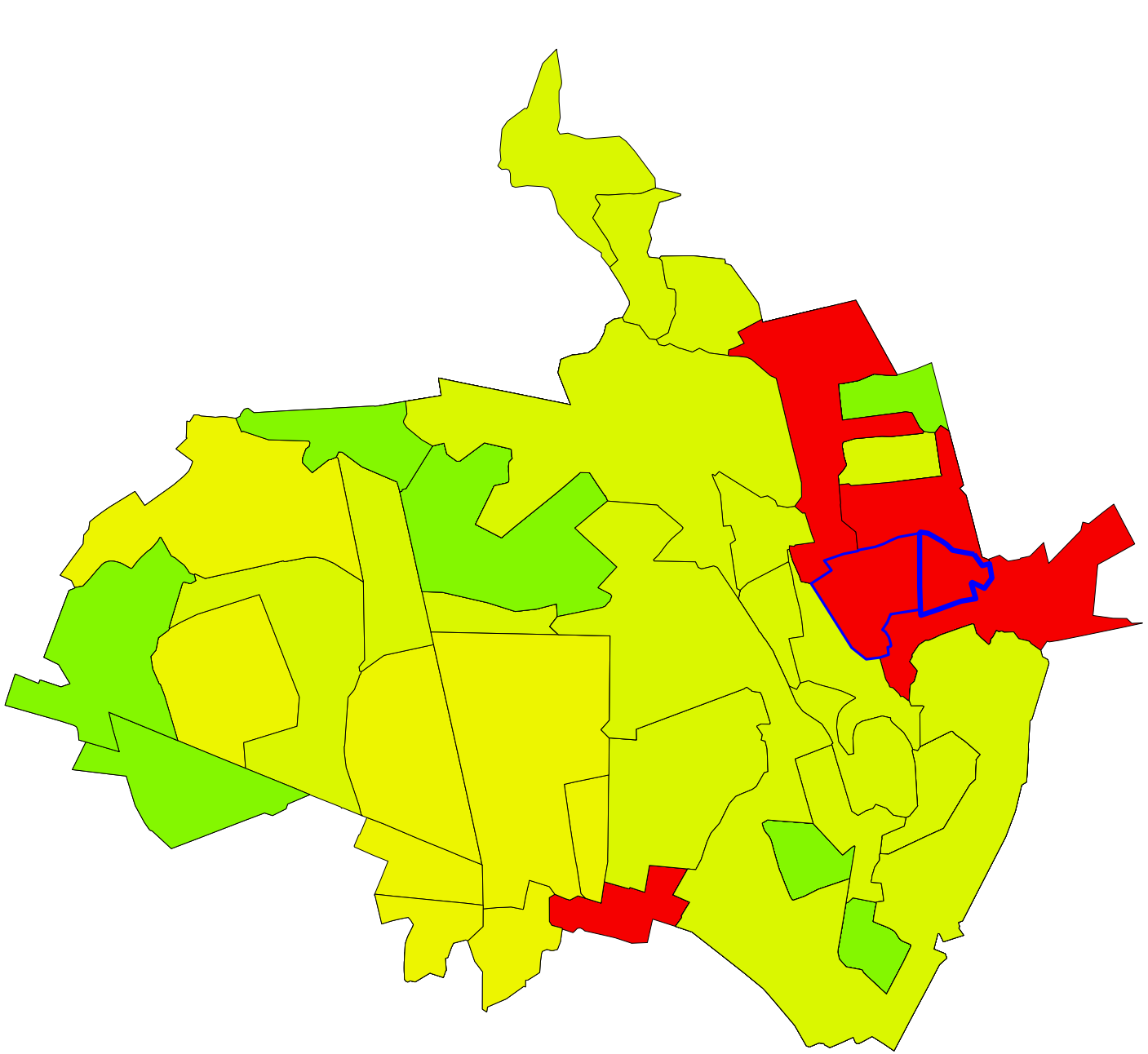


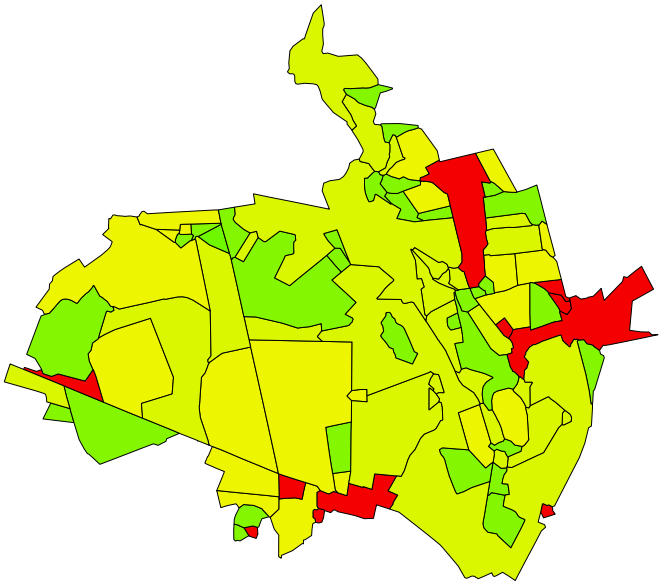


source: 92 patches

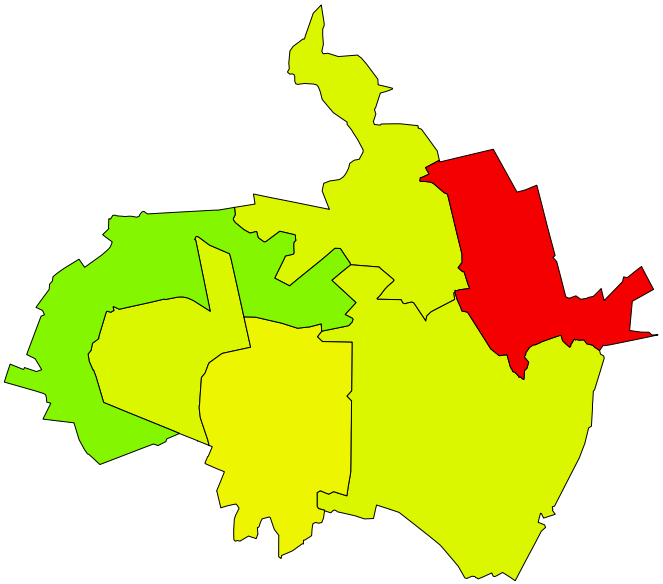


target: 6 patches

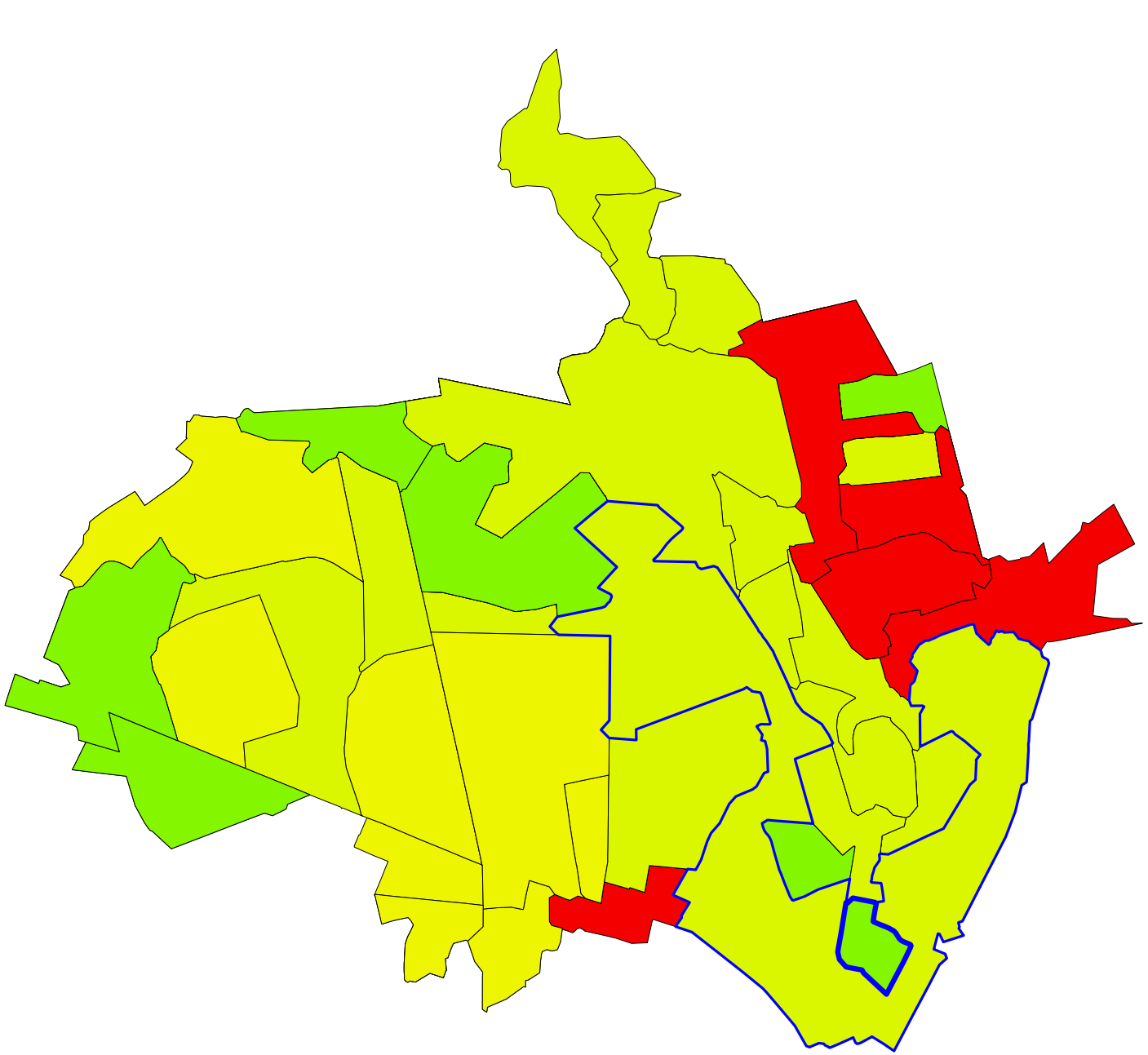


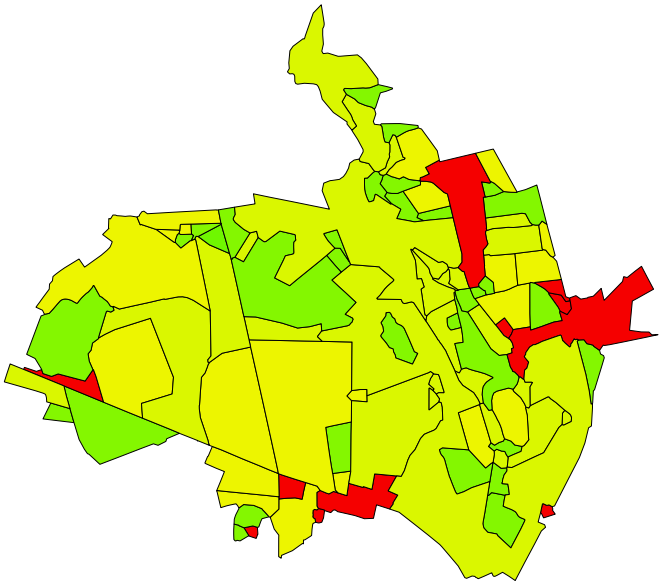


source: 92 patches

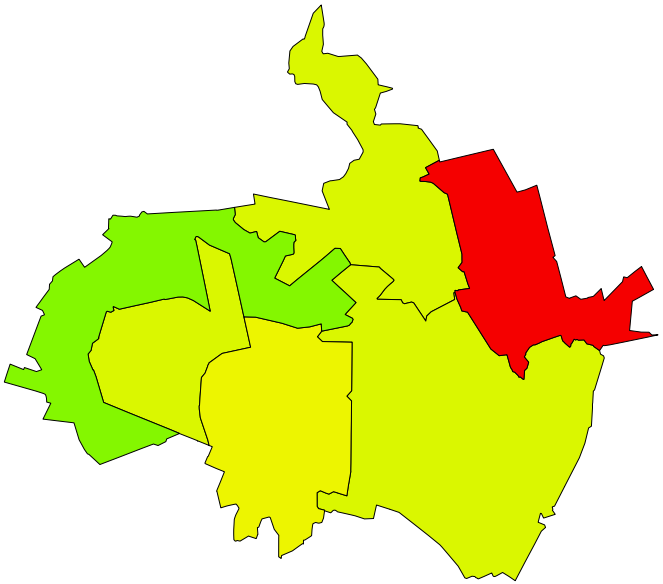


target: 6 patches

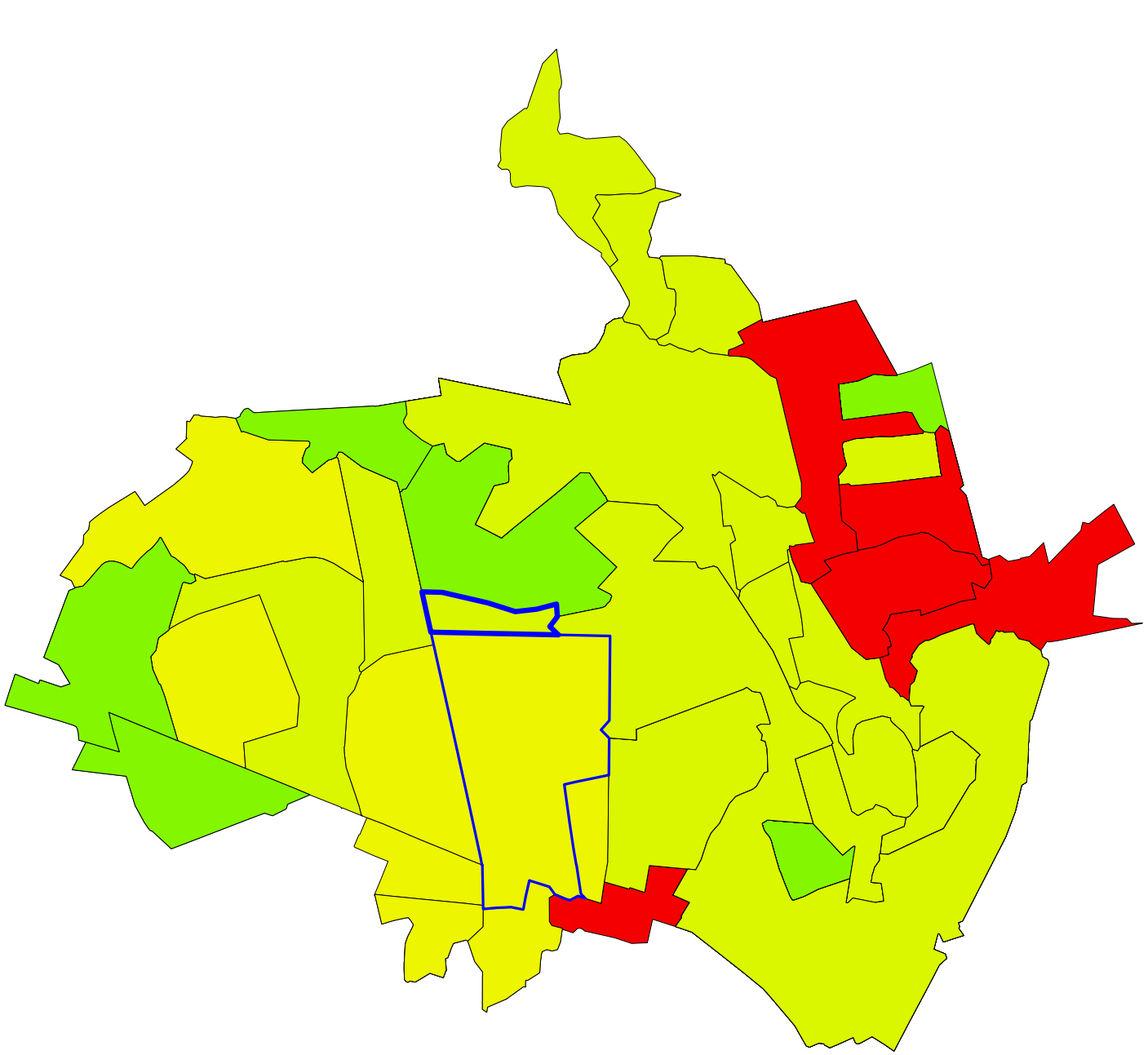


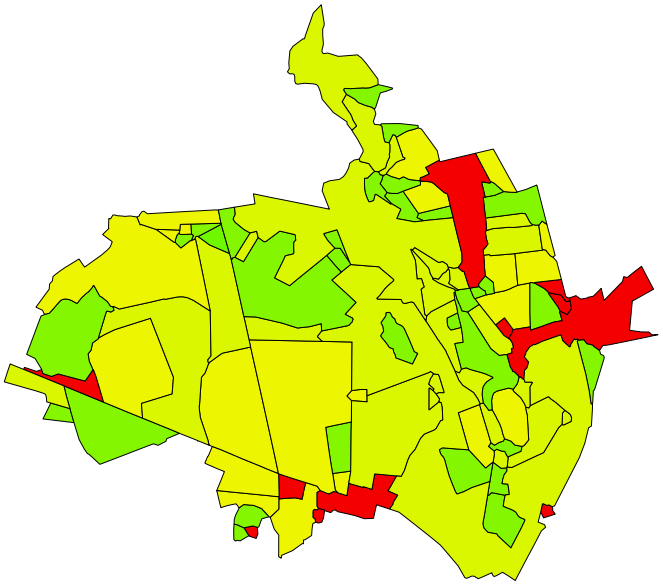


source: 92 patches

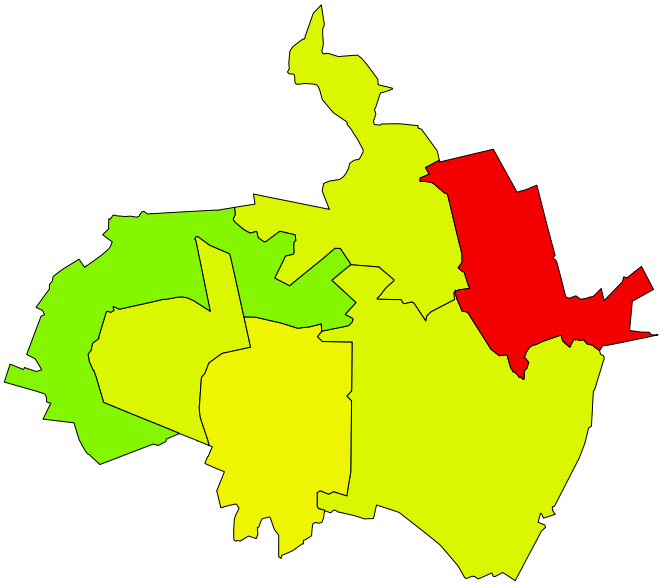


target: 6 patches

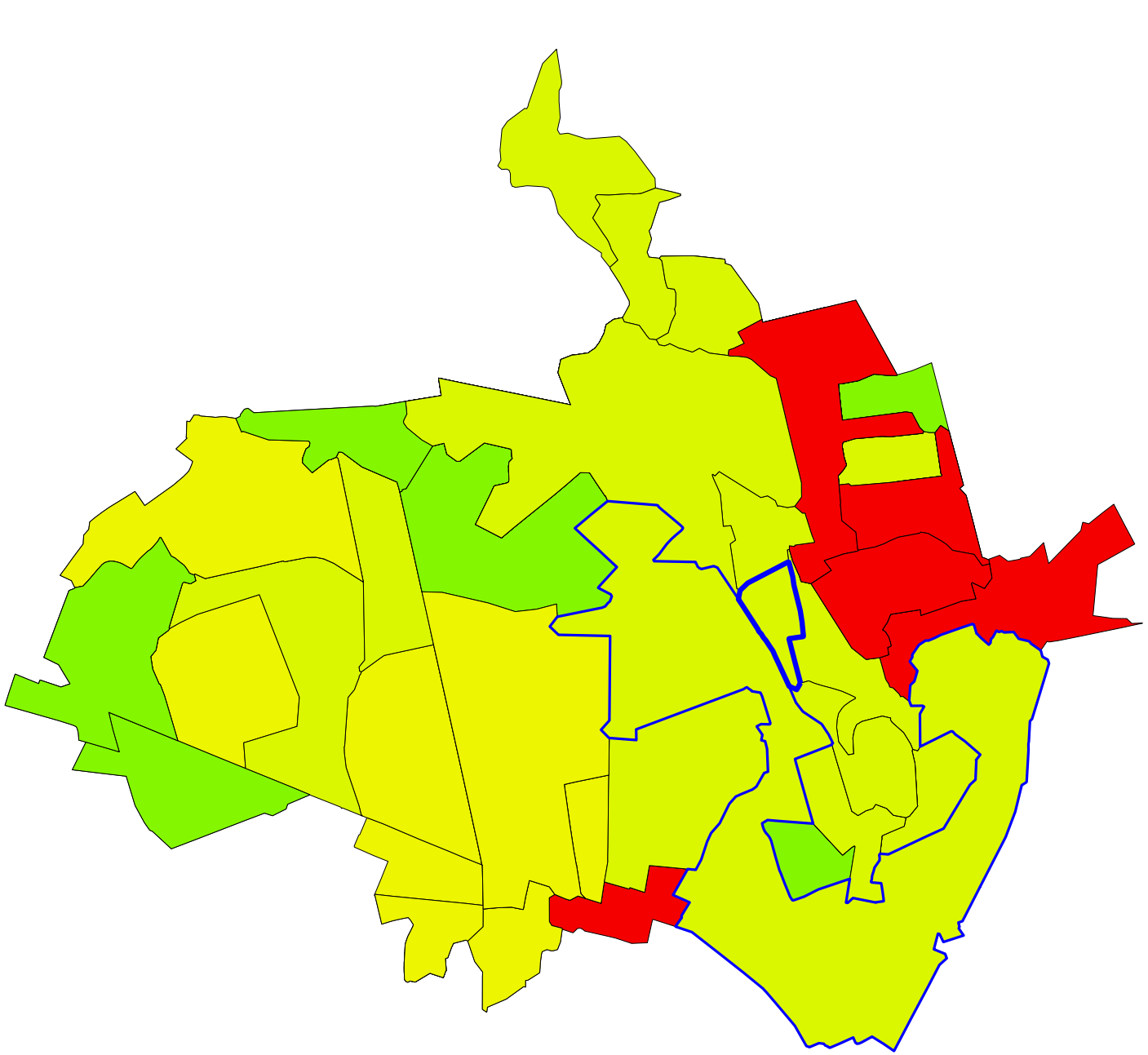


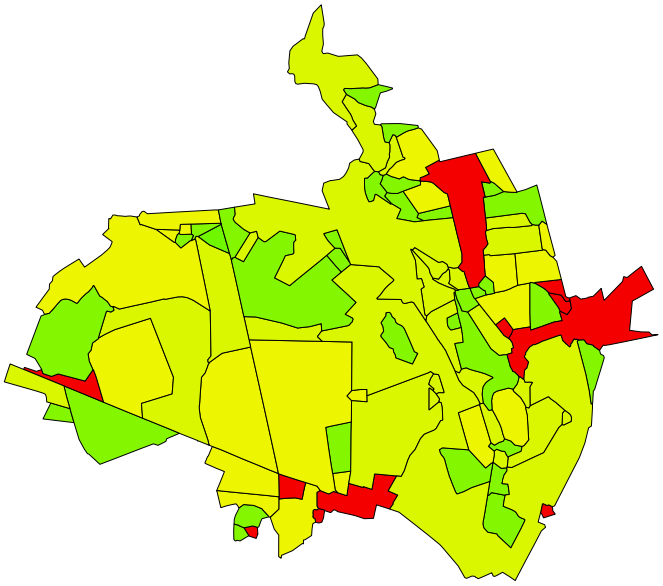


source: 92 patches

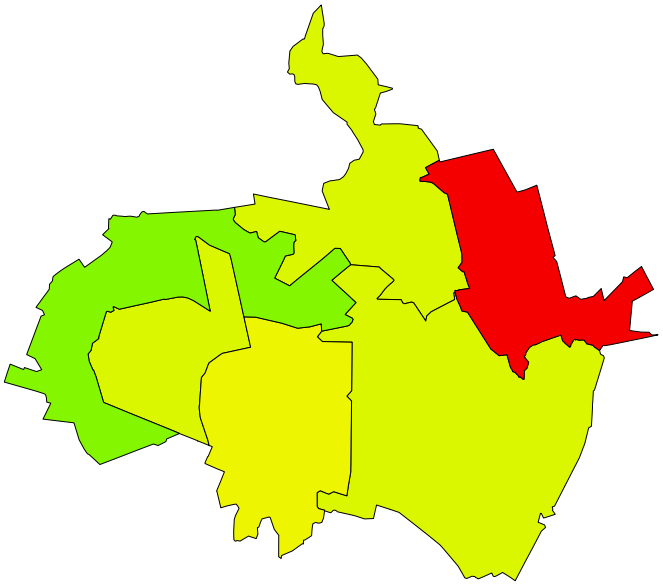


target: 6 patches

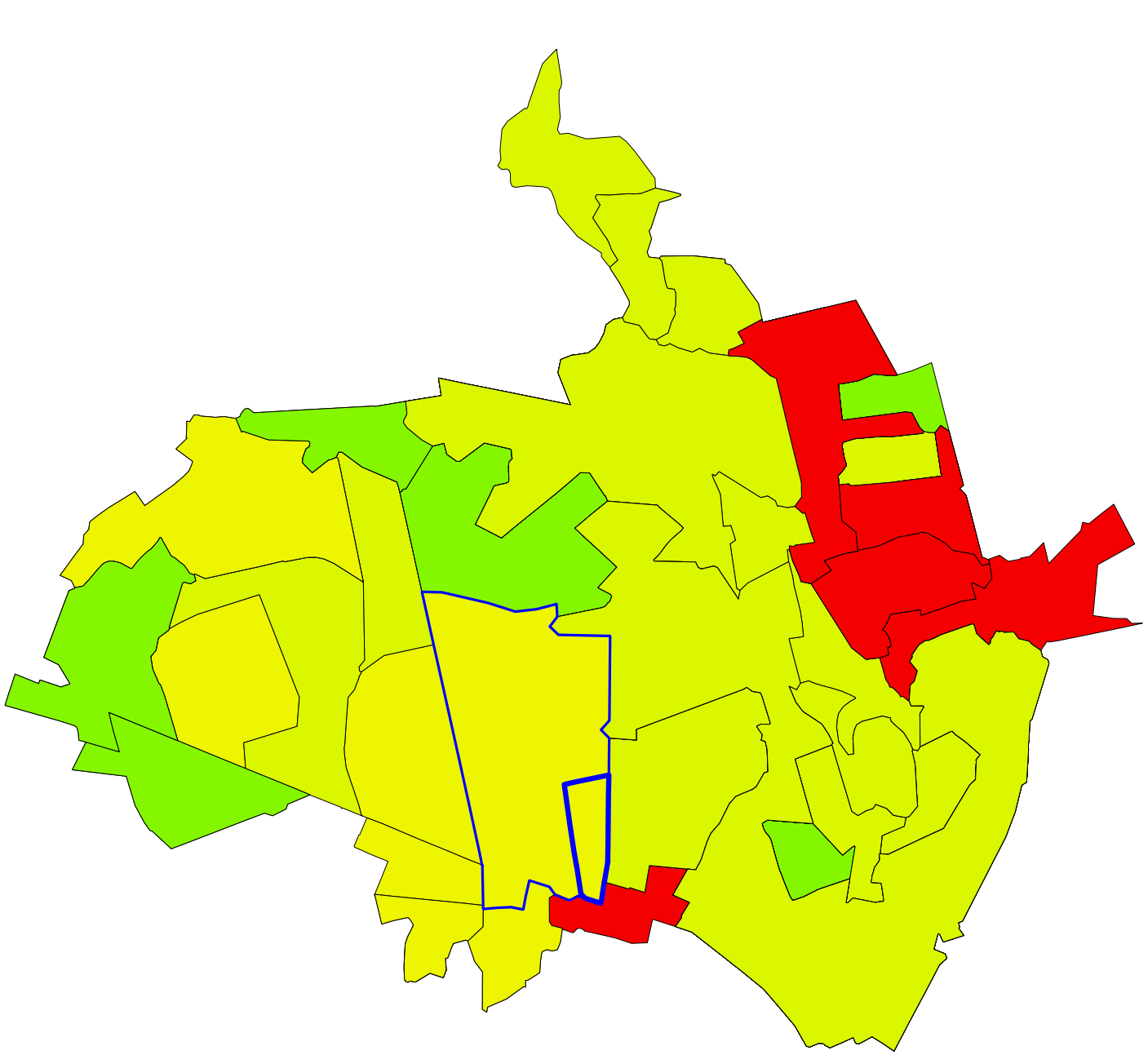


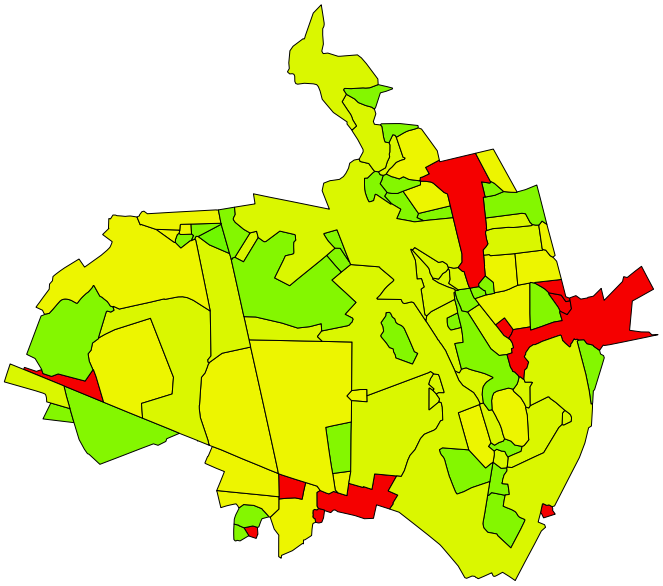


source: 92 patches

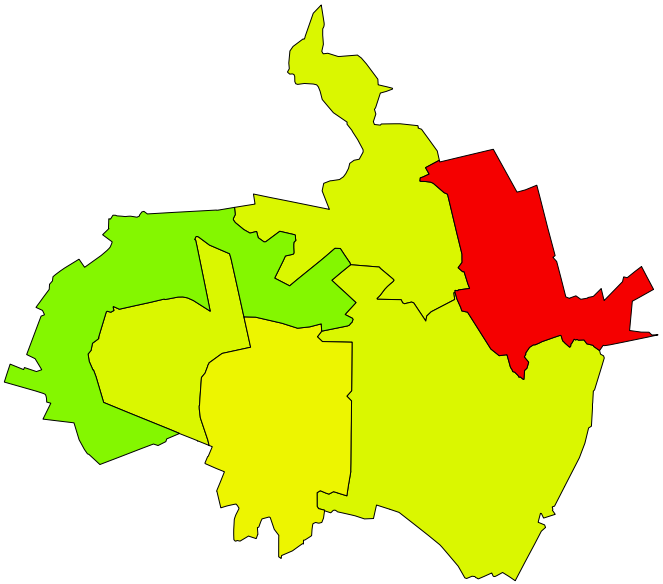


target: 6 patches

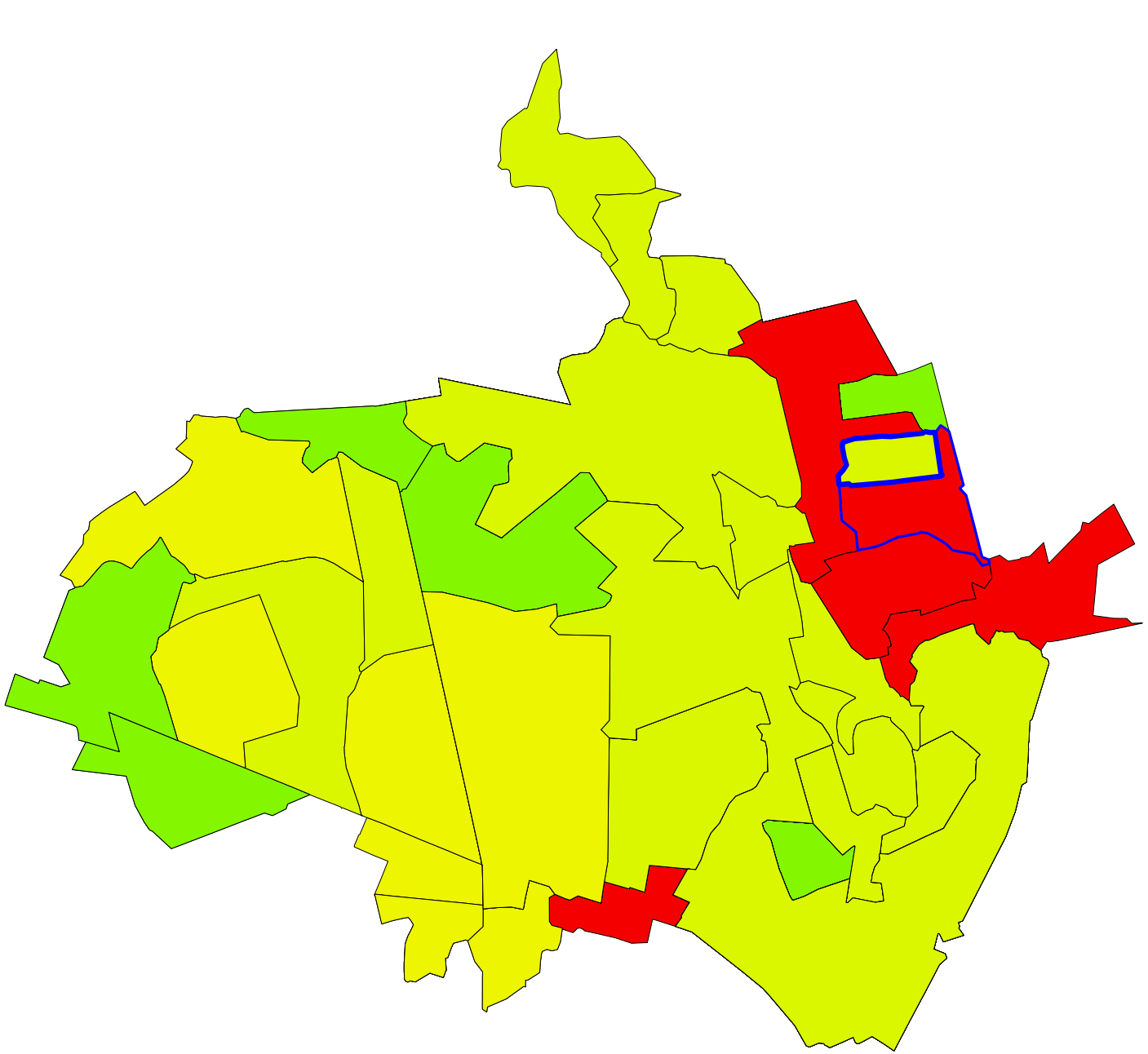


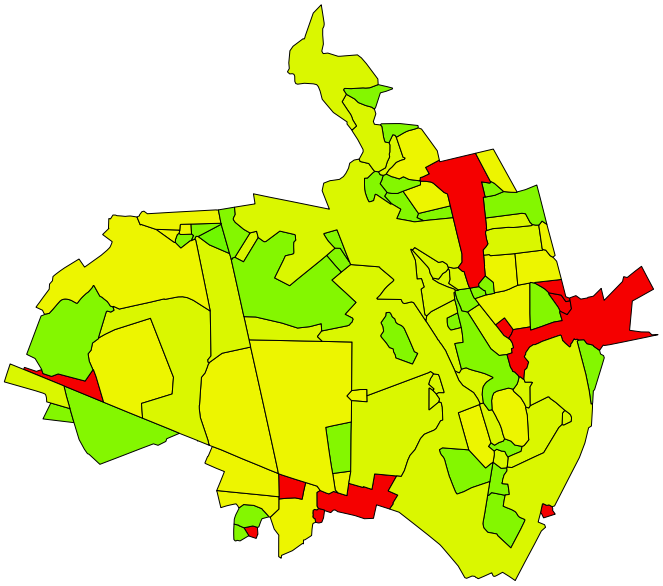


source: 92 patches

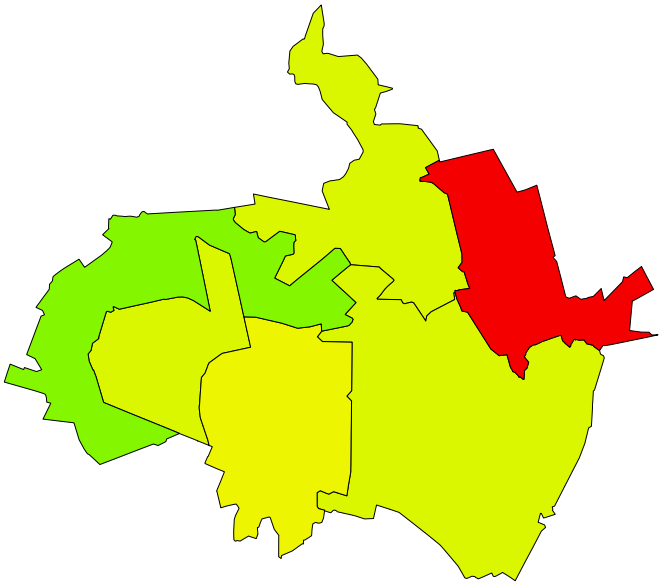


target: 6 patches

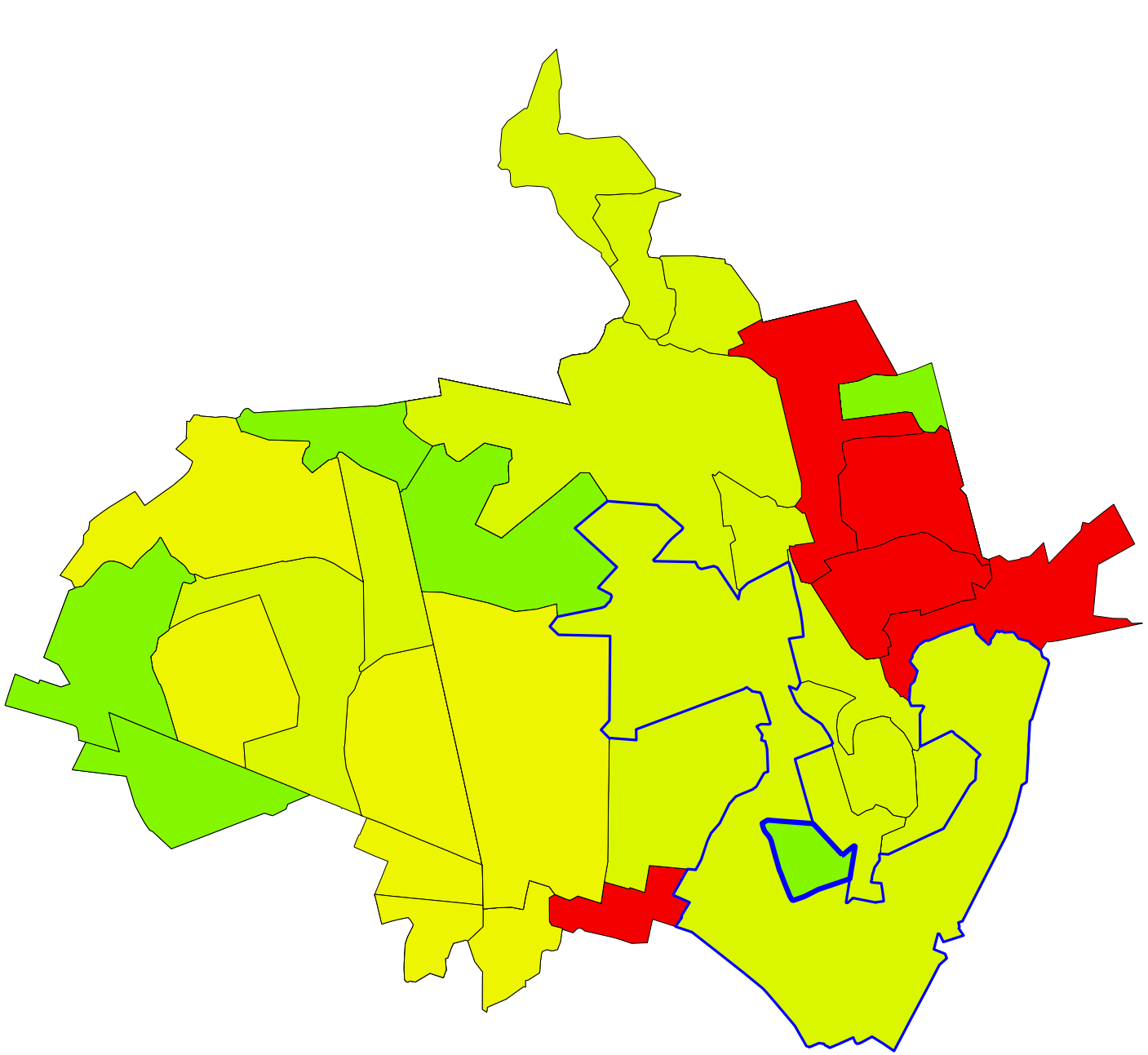


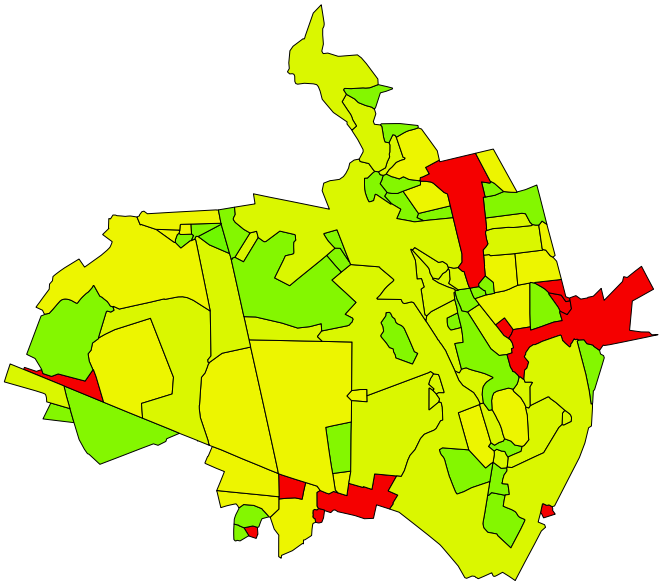


source: 92 patches

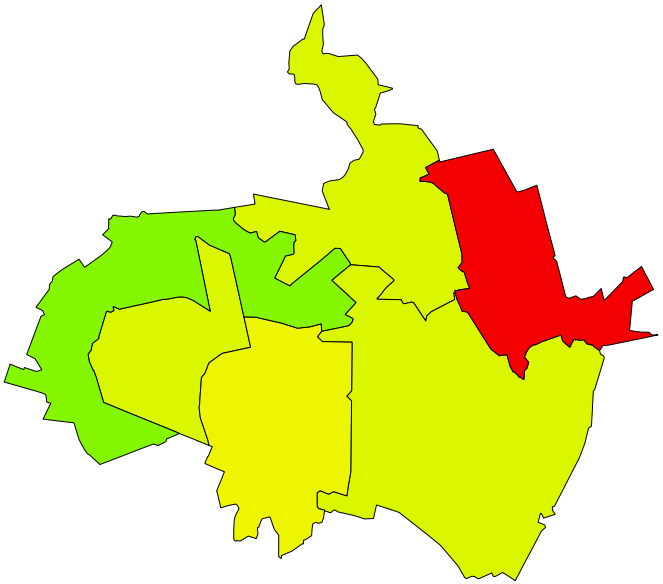


target: 6 patches

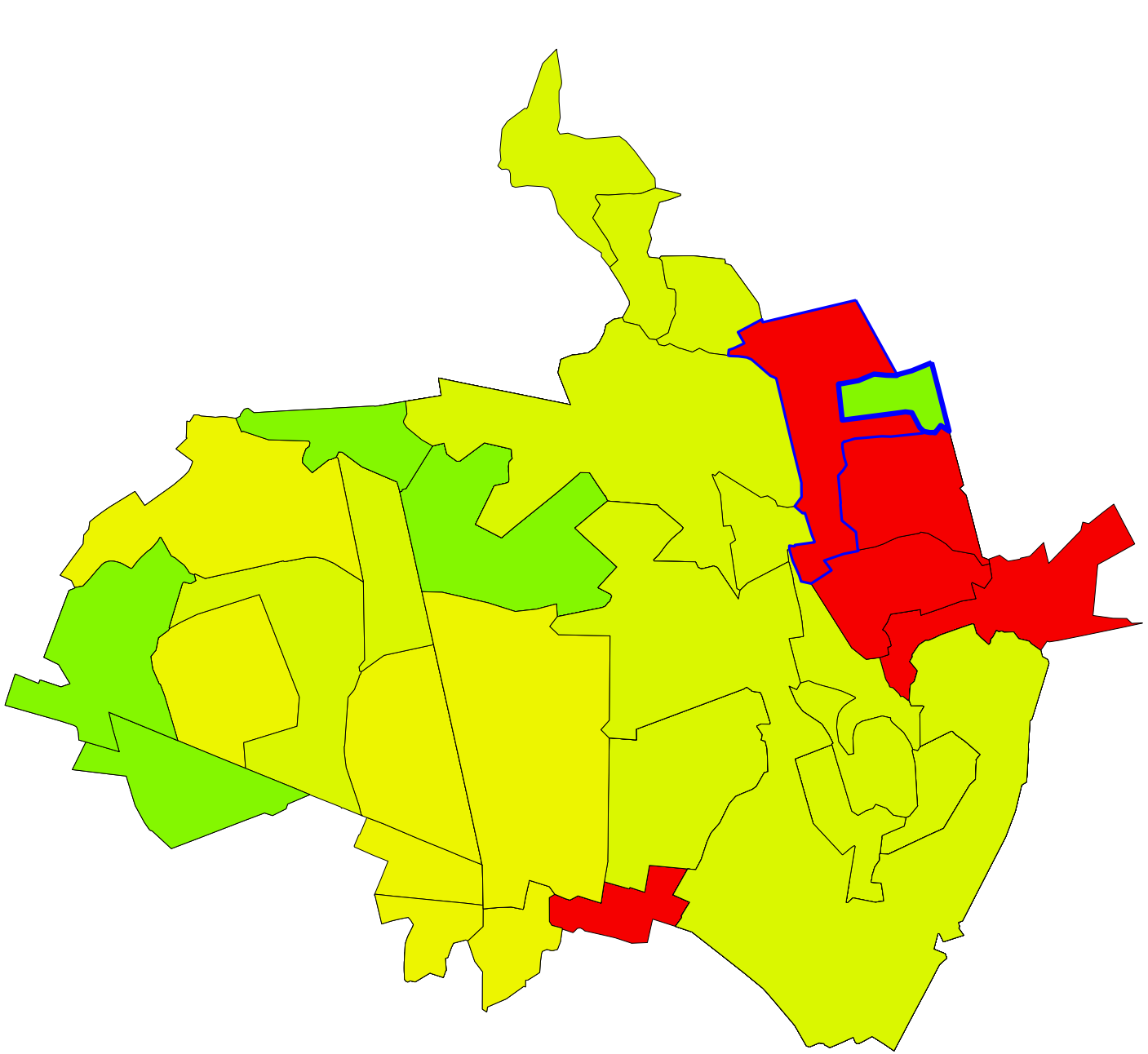


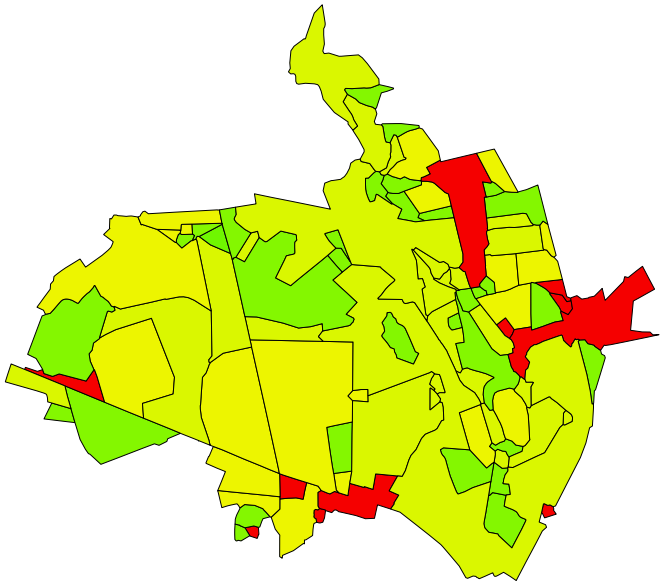


source: 92 patches

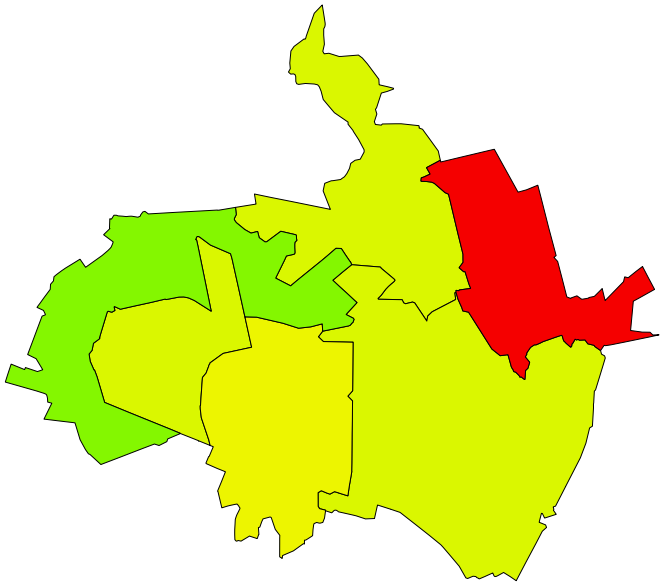


target: 6 patches

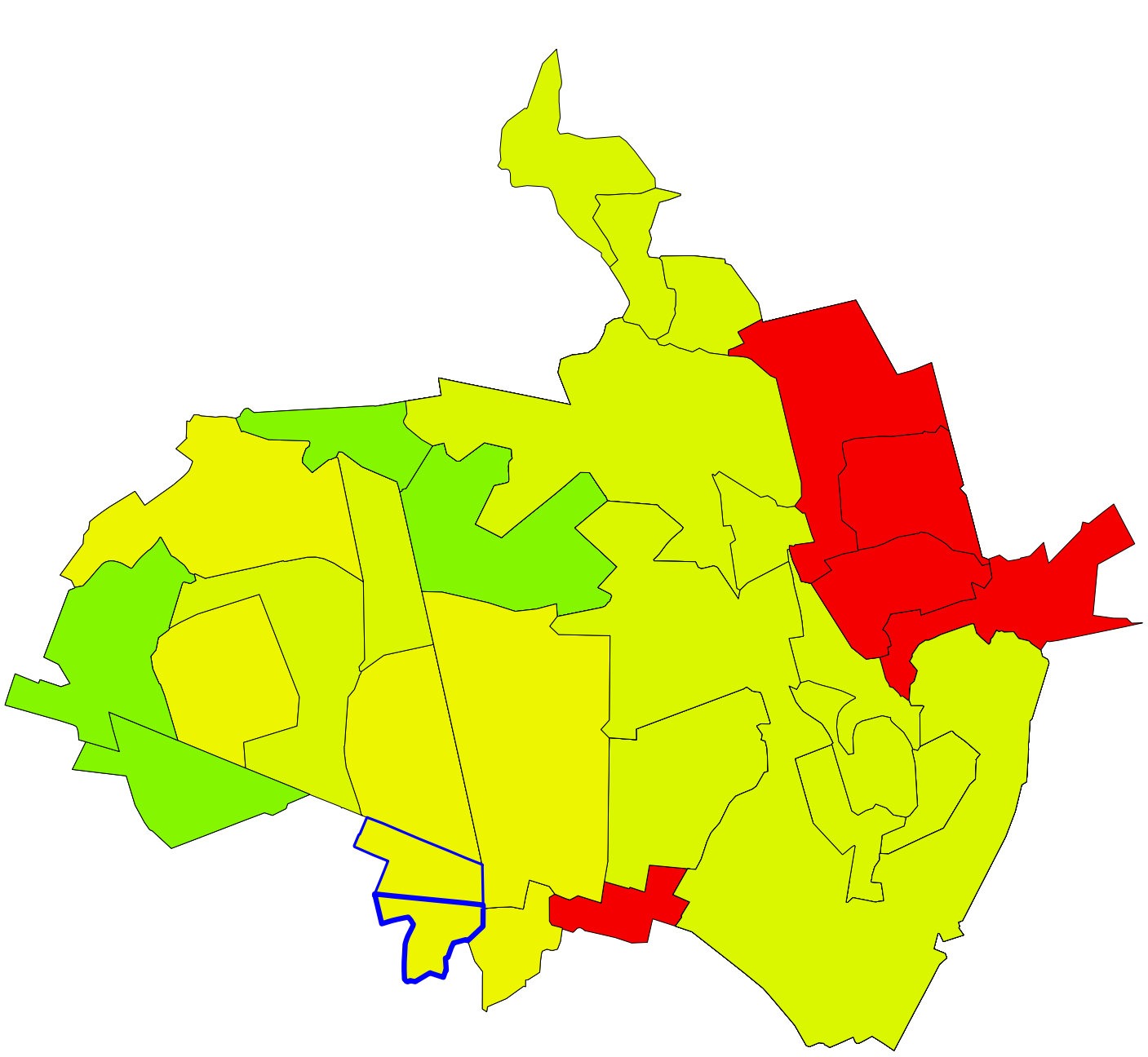


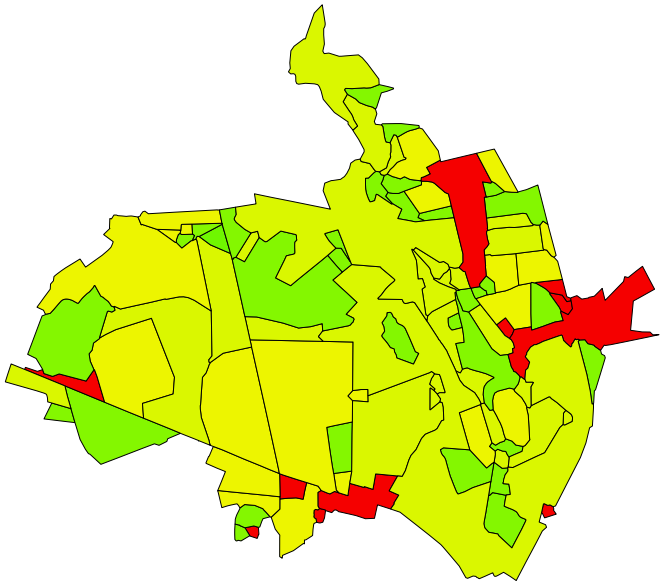


source: 92 patches

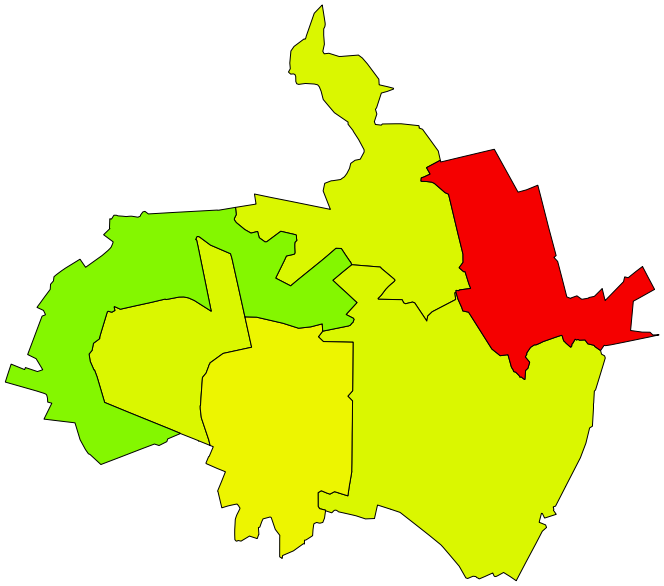


target: 6 patches

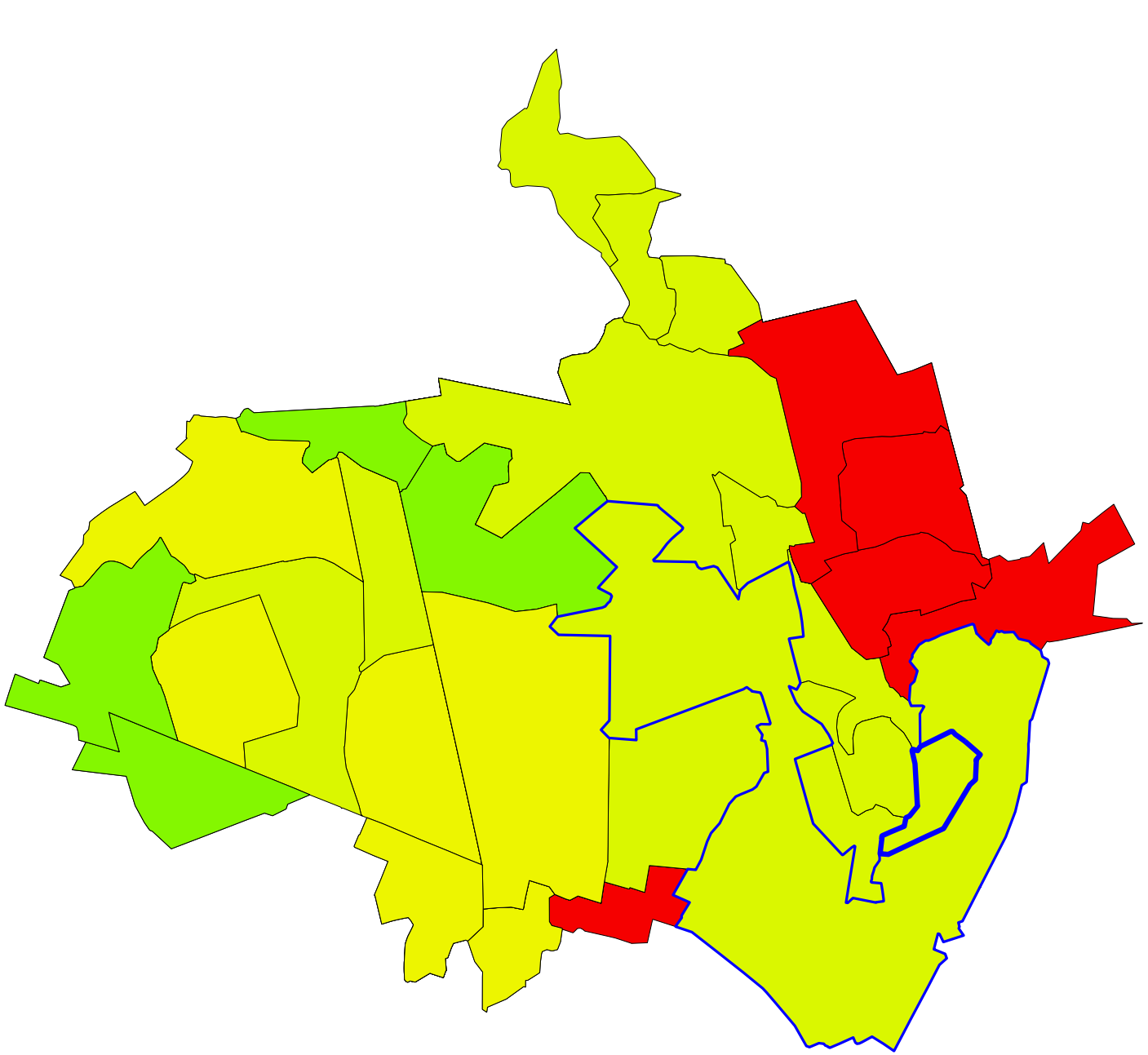


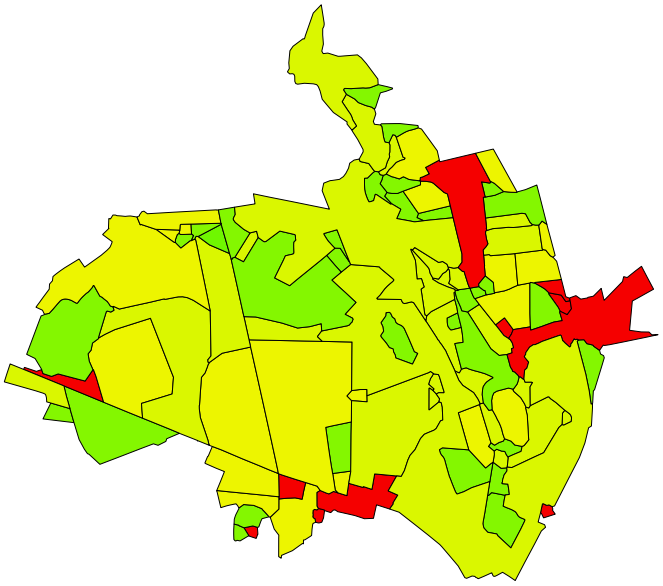


source: 92 patches

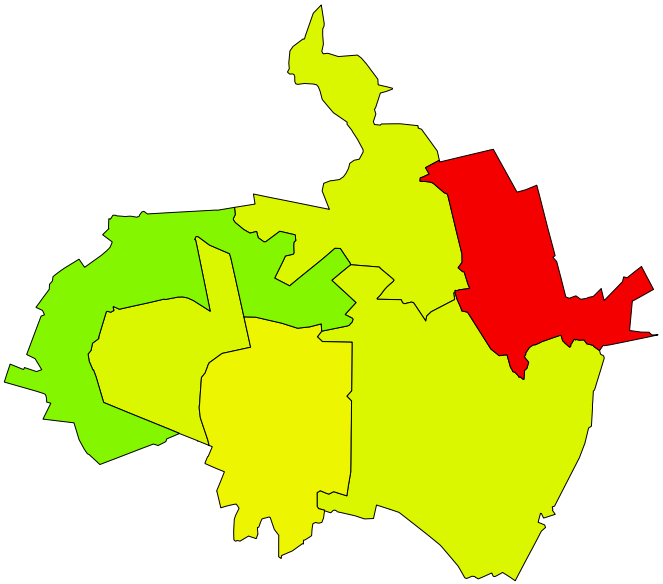


target: 6 patches

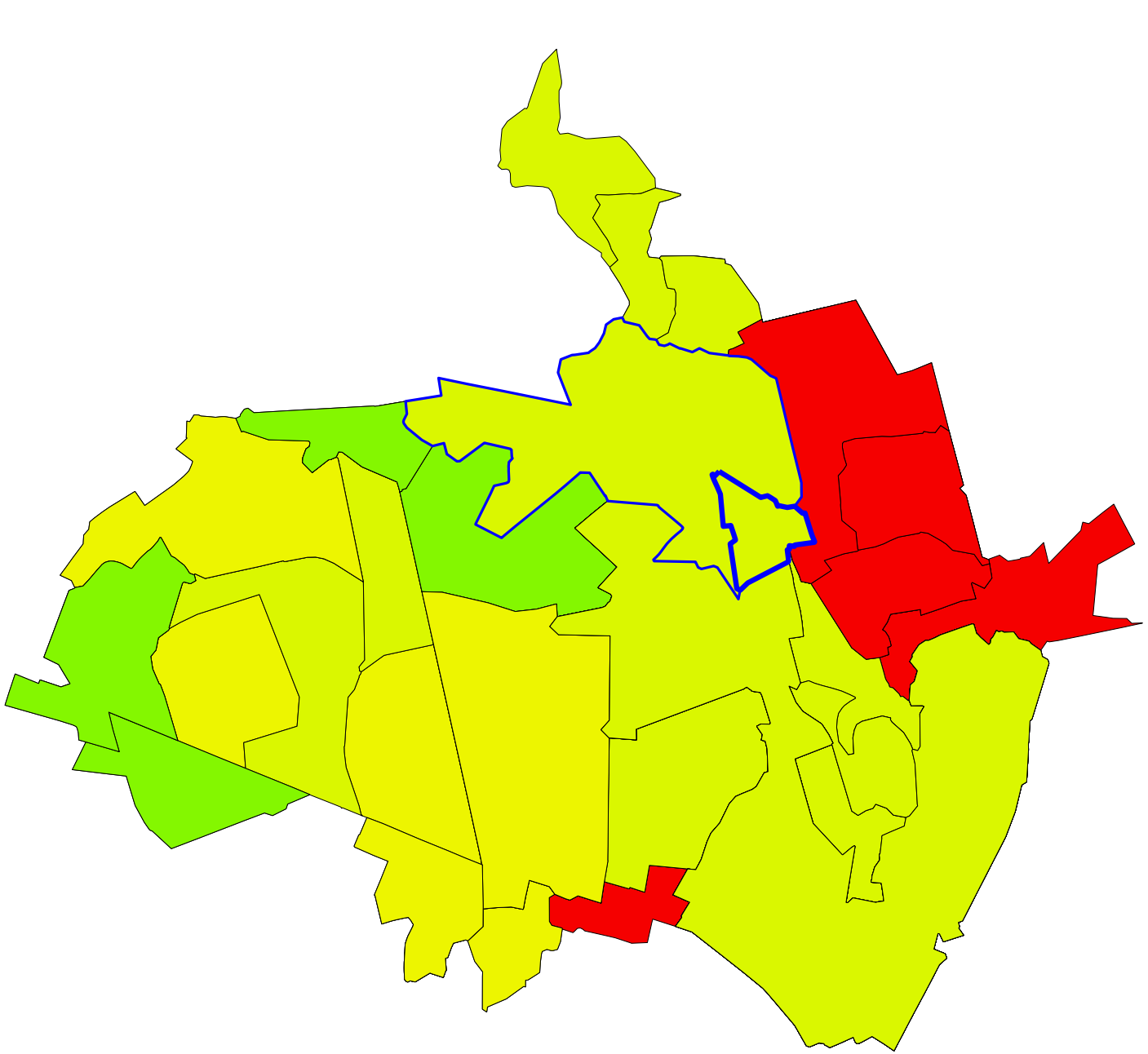


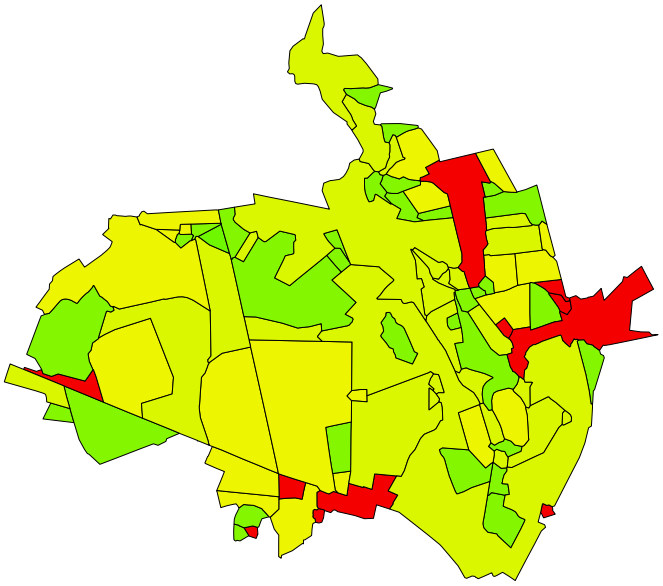


source: 92 patches

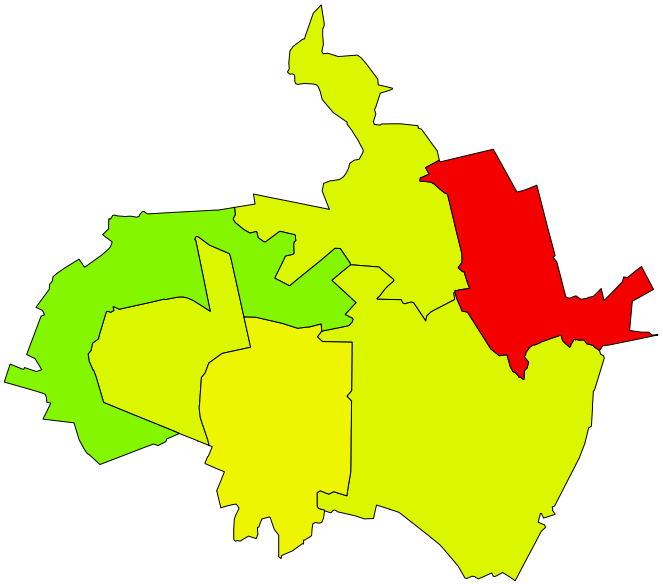


target: 6 patches

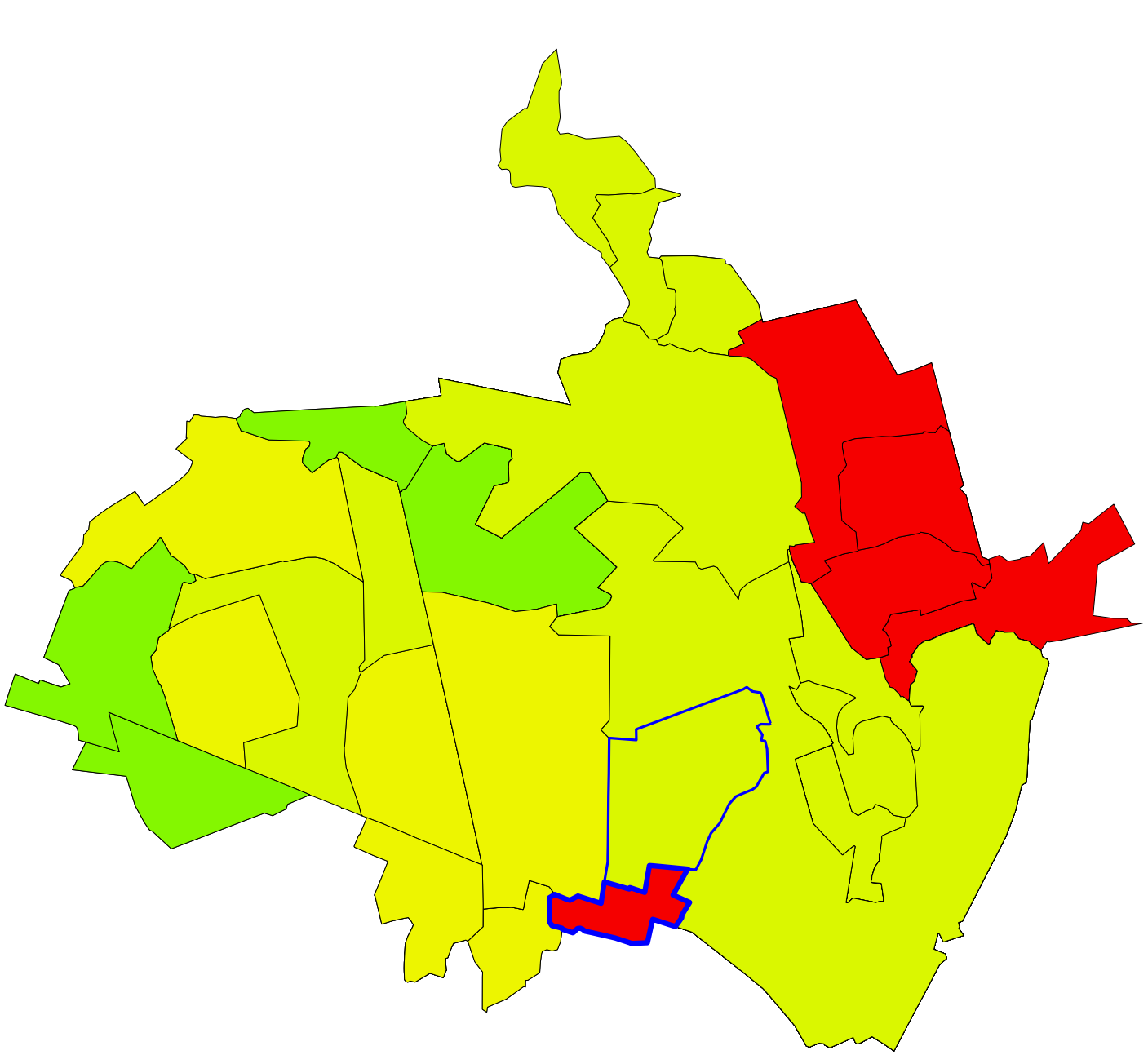


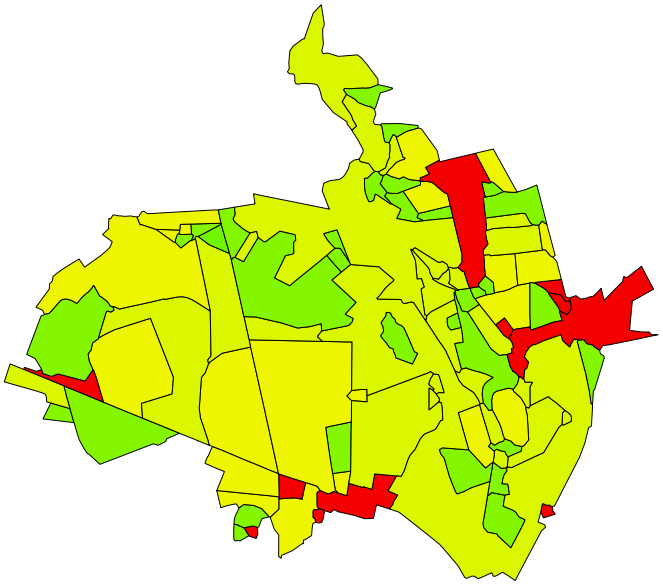


source: 92 patches

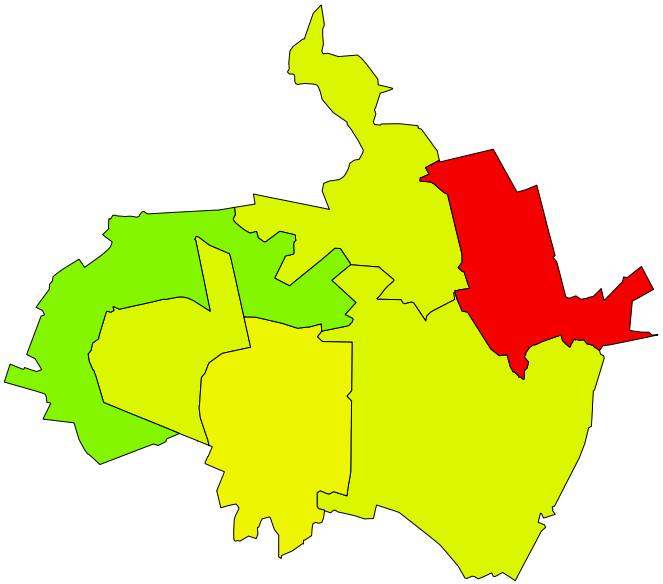


target: 6 patches

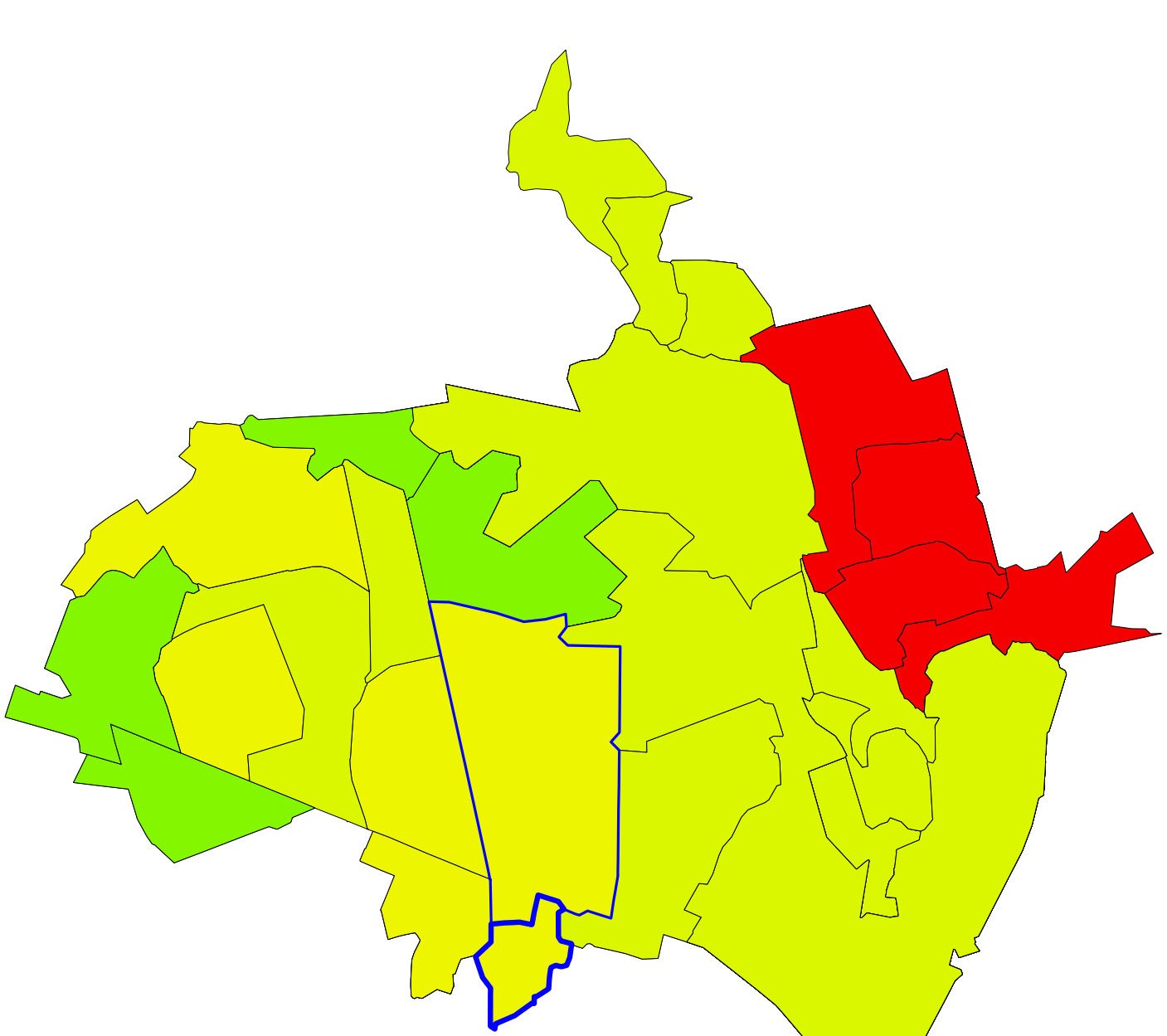


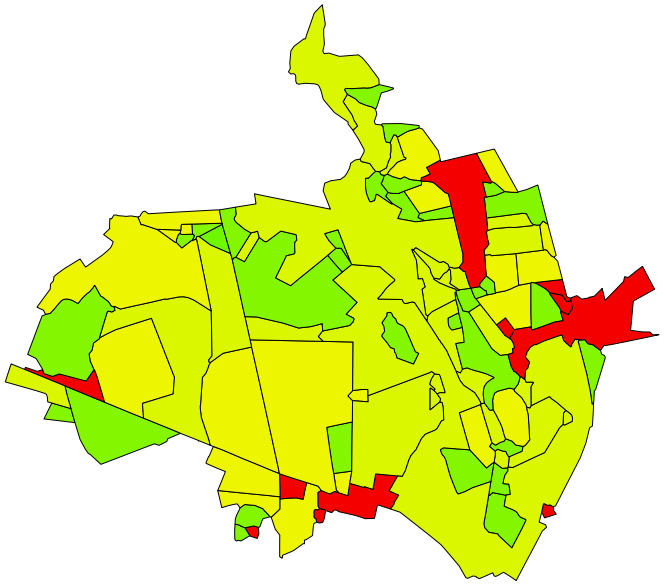


source: 92 patches

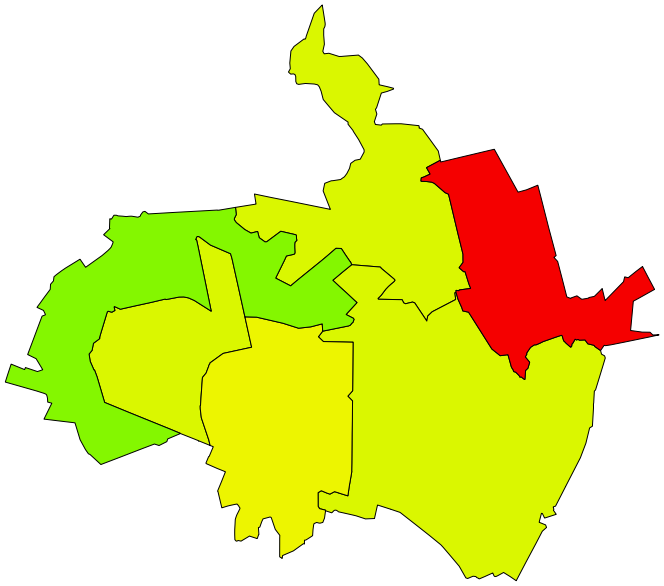


target: 6 patches

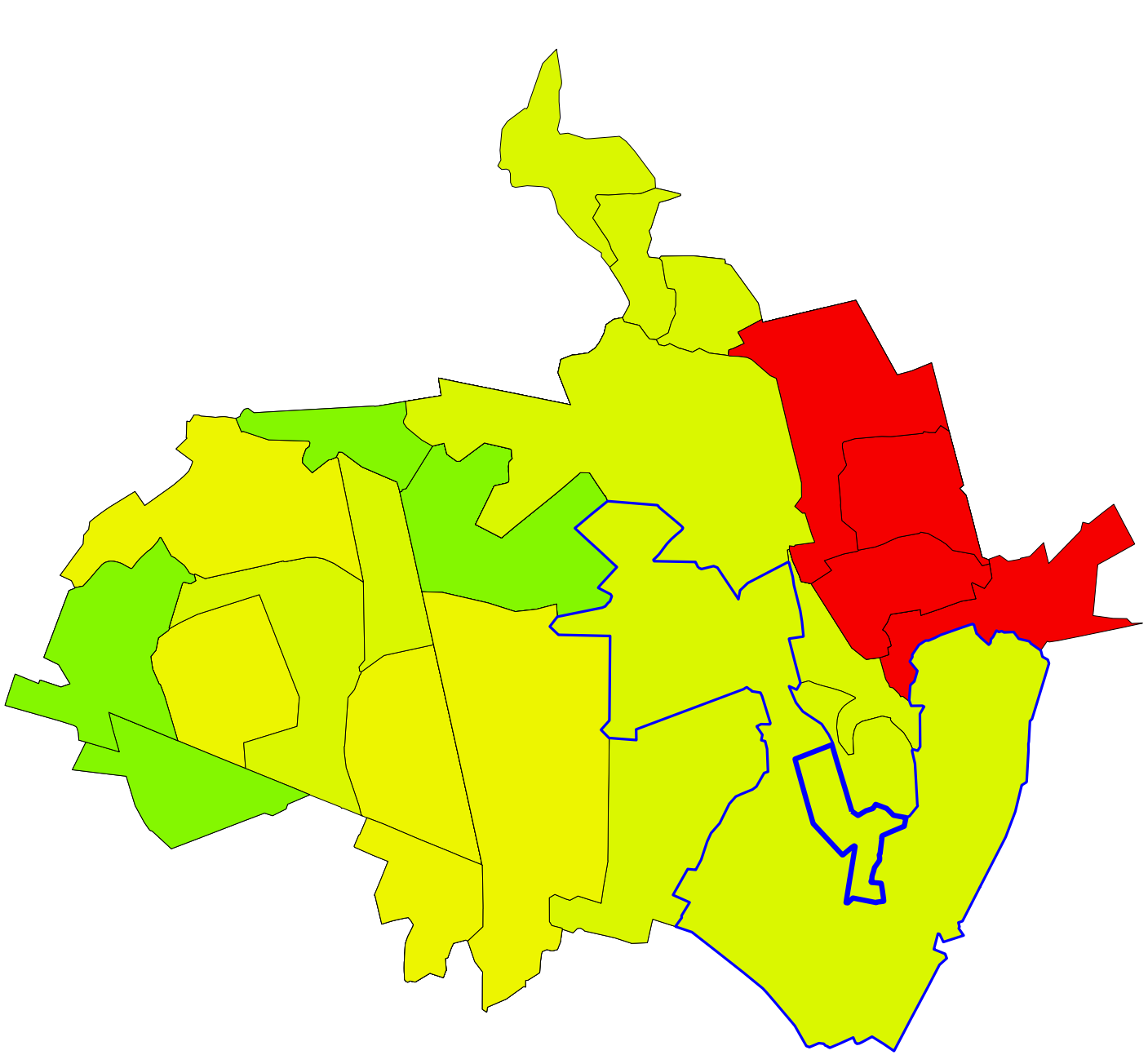


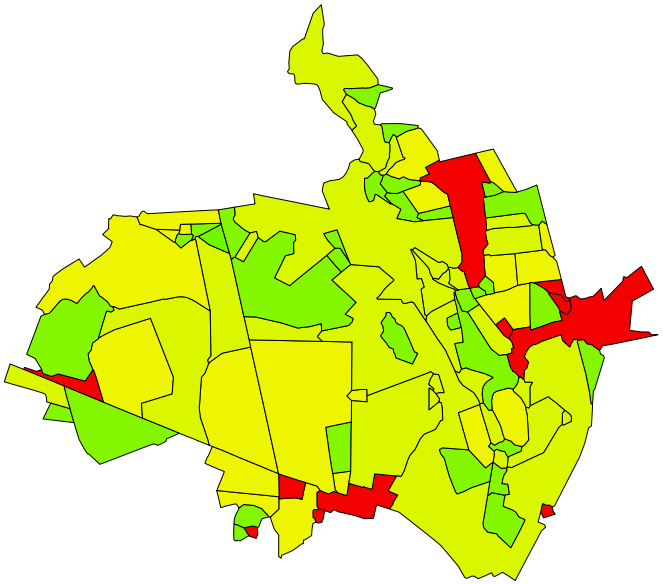


source: 92 patches

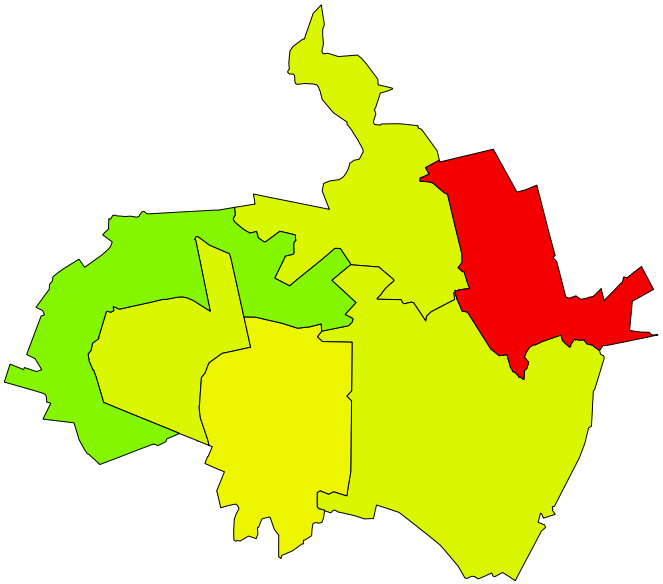


target: 6 patches

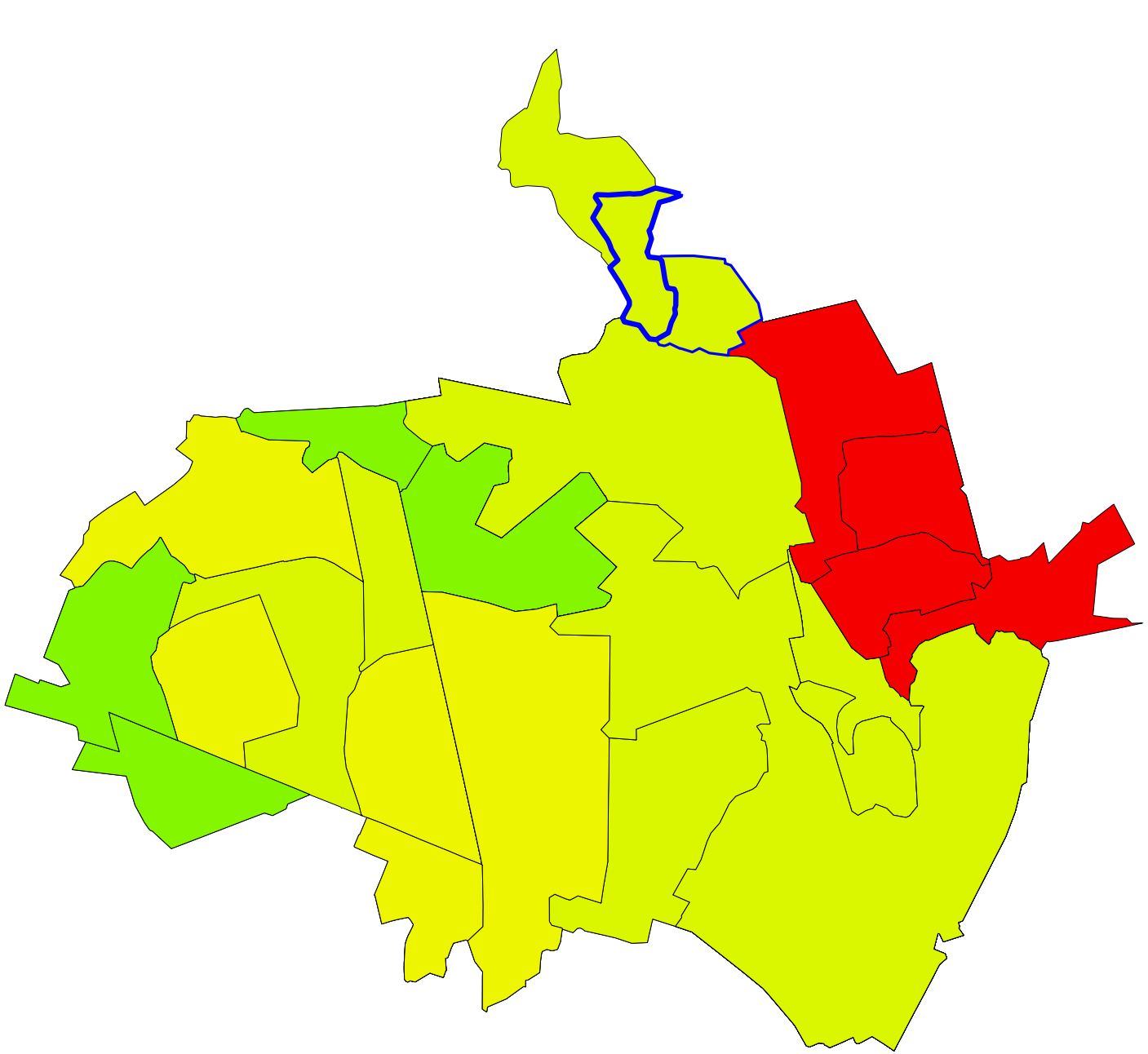


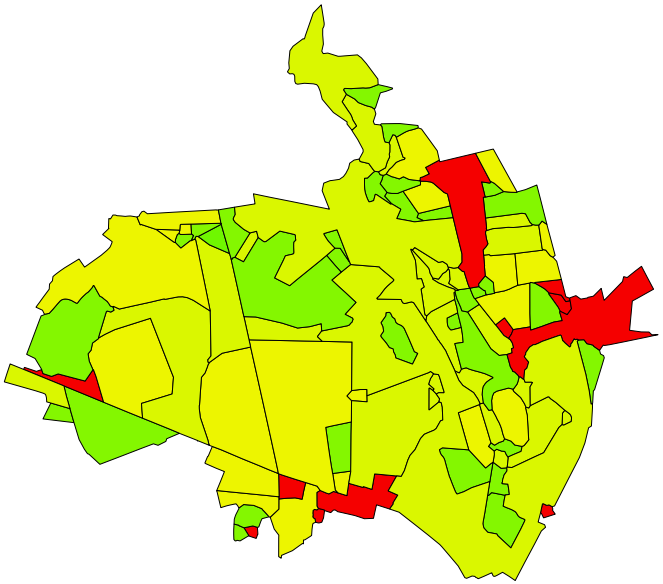


source: 92 patches

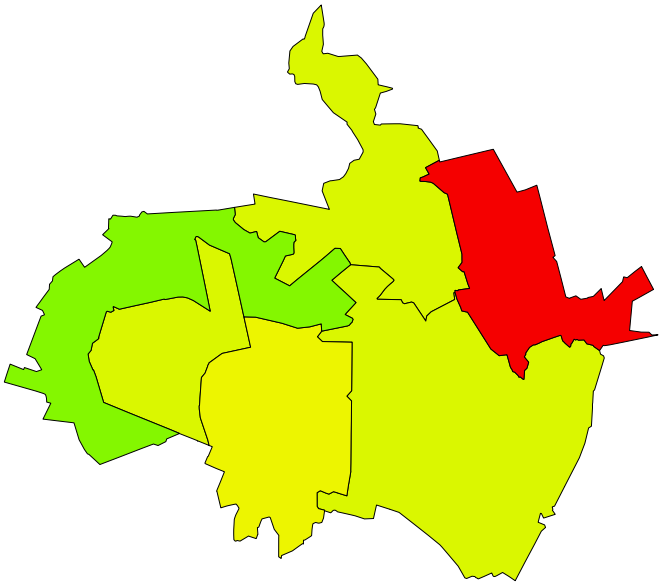


target: 6 patches

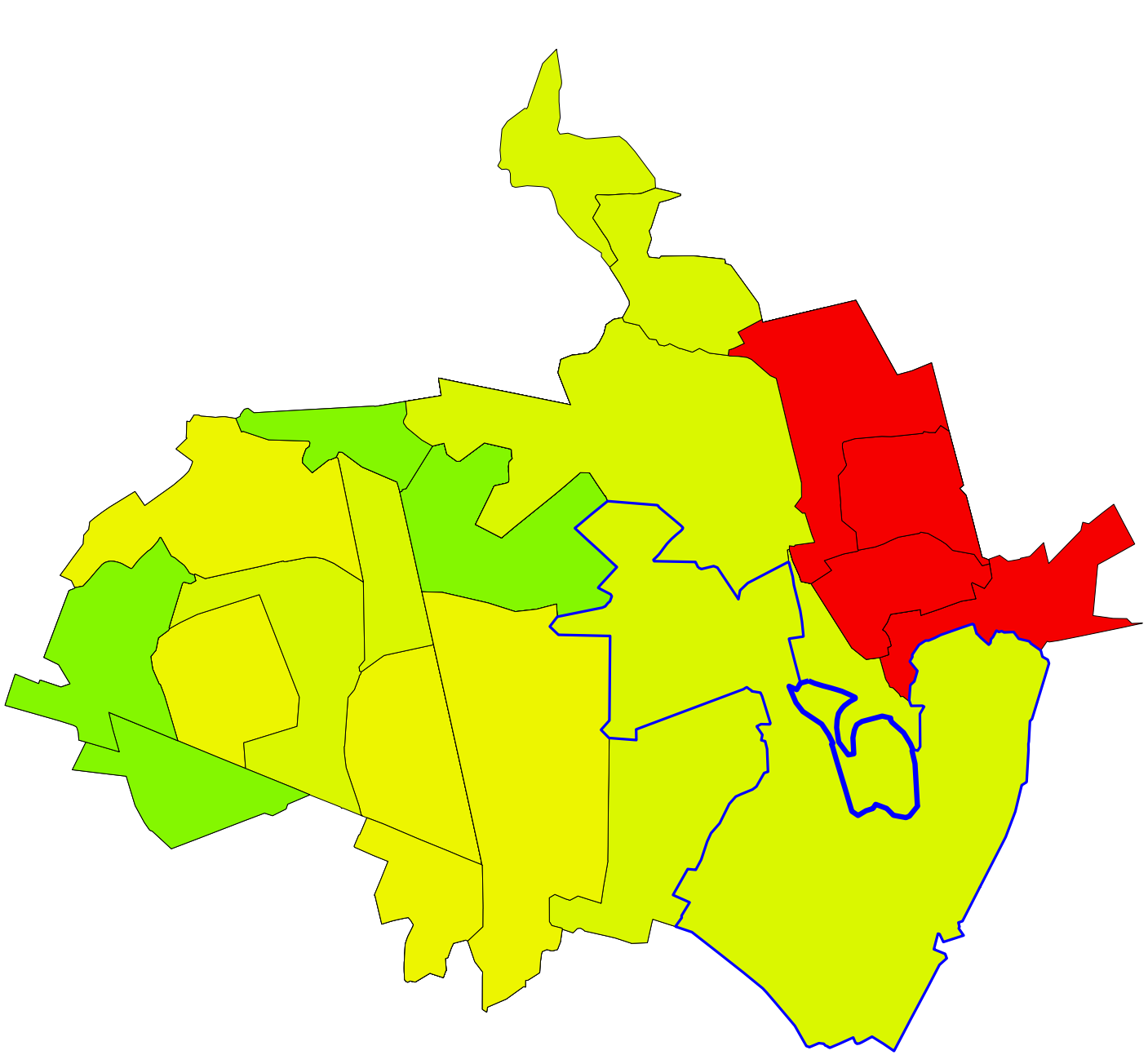


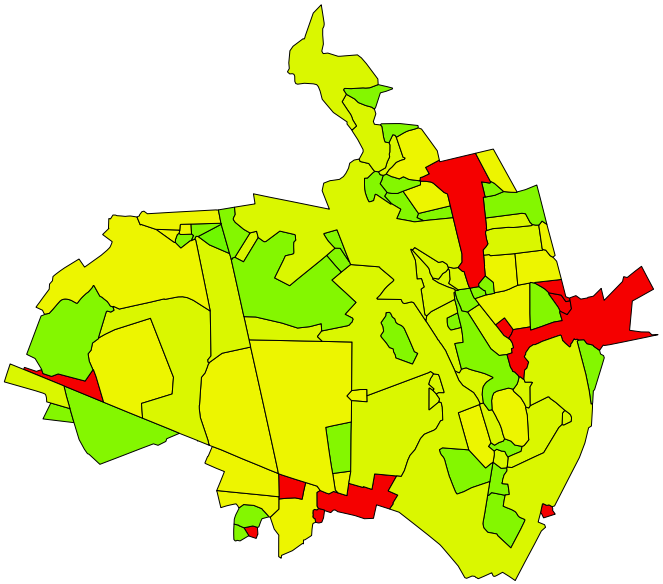


source: 92 patches

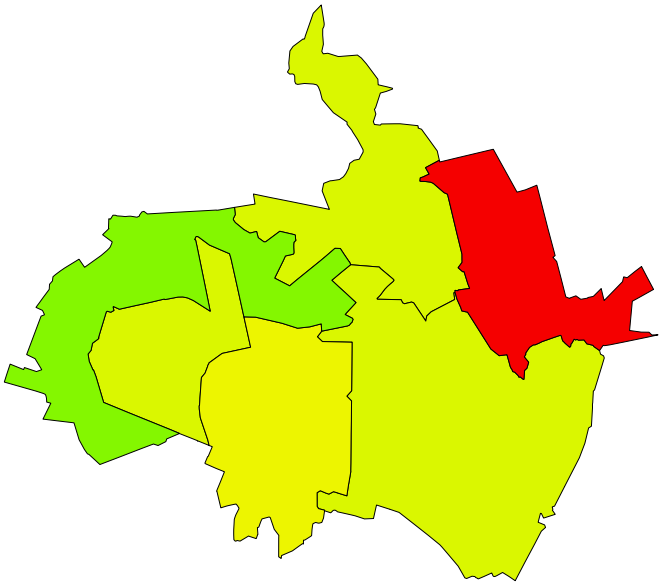


target: 6 patches

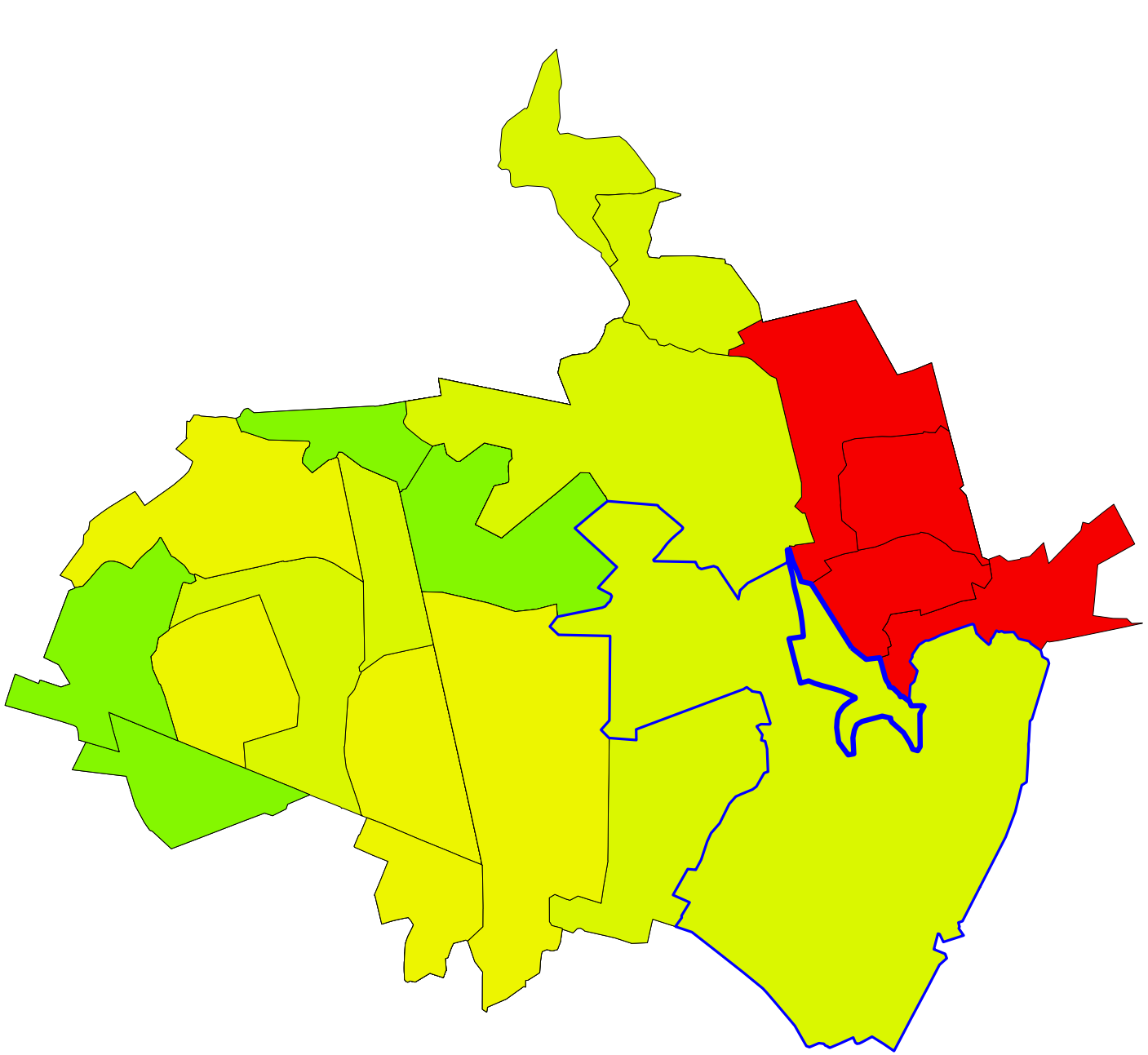


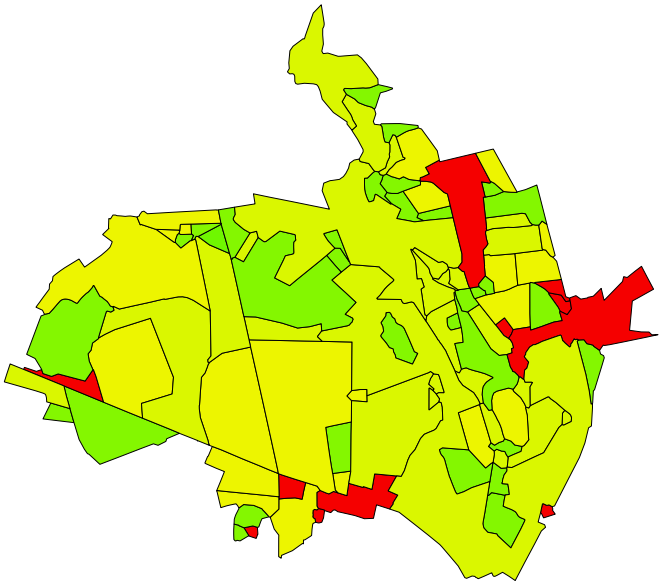


source: 92 patches

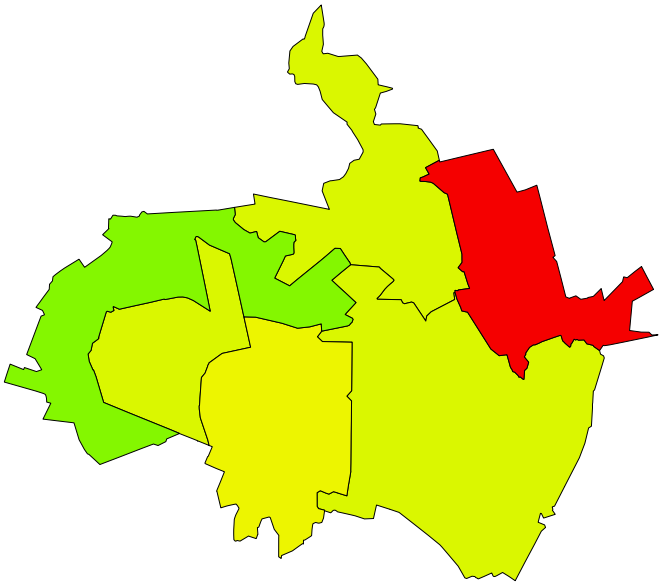


target: 6 patches

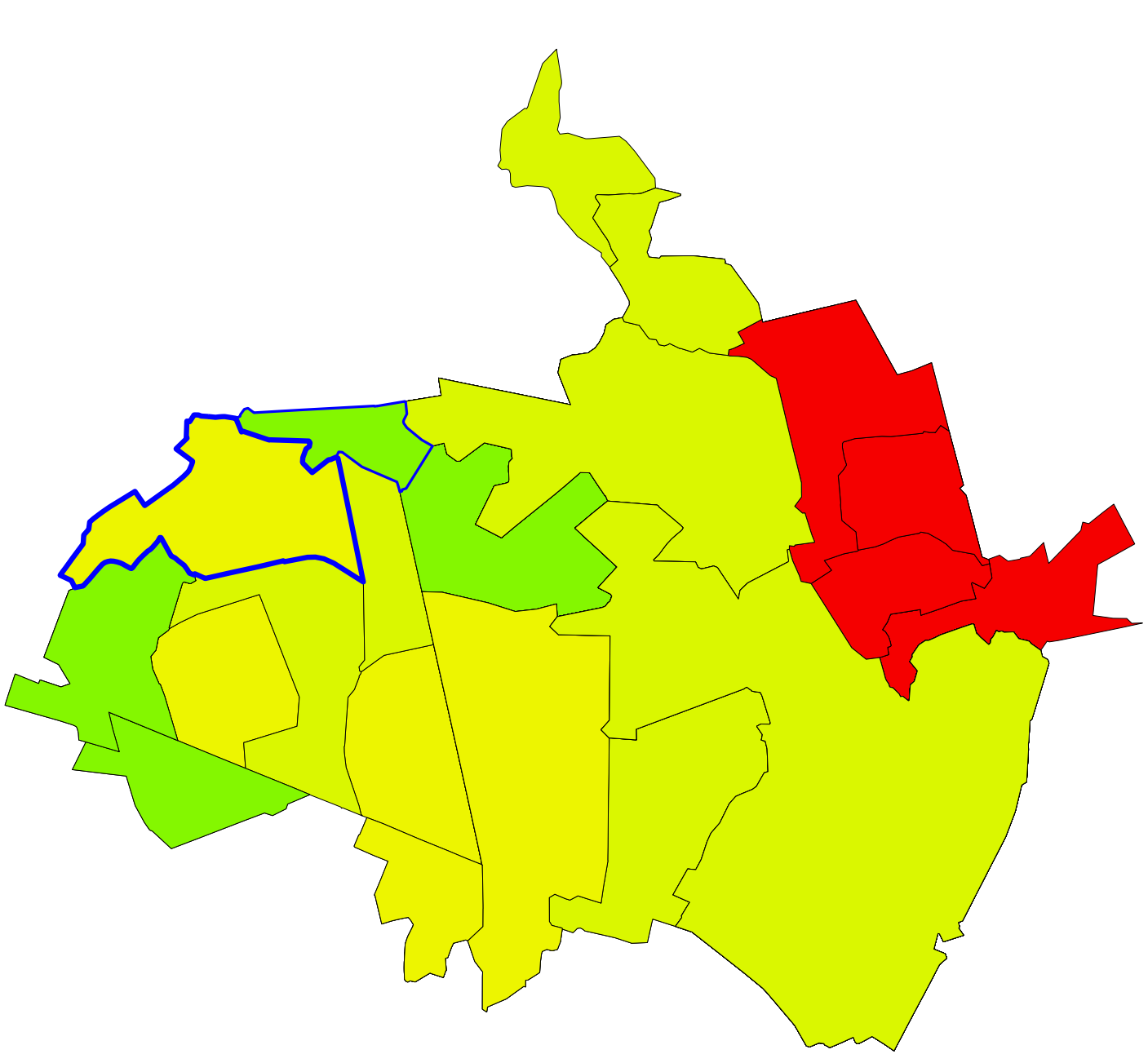


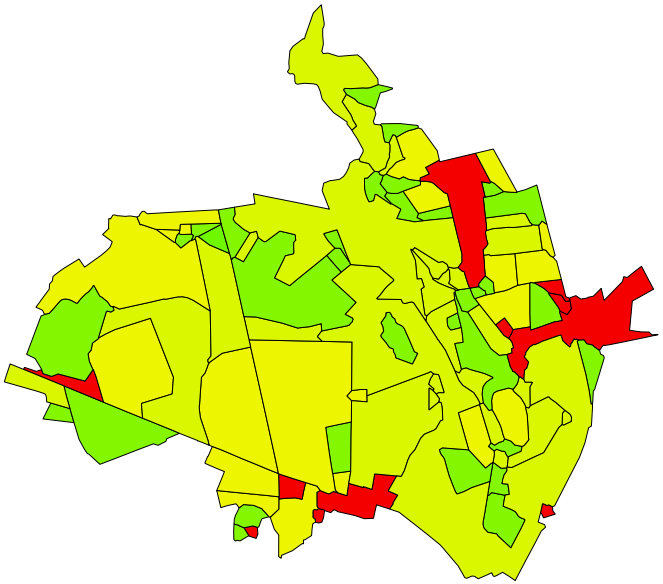


source: 92 patches

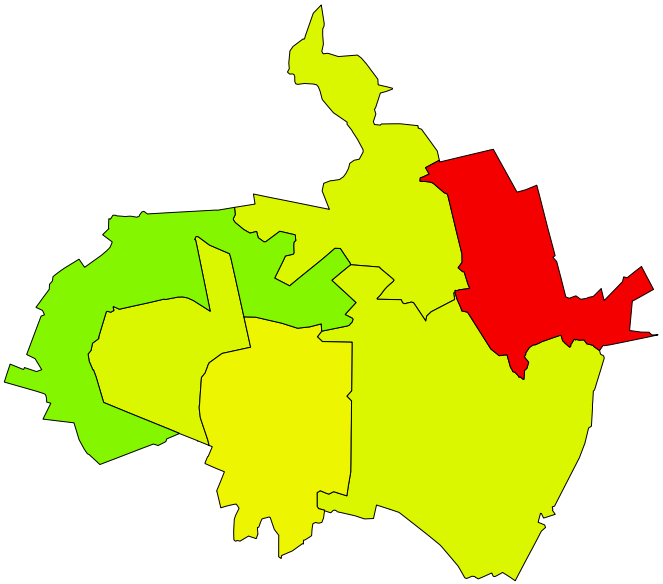


target: 6 patches

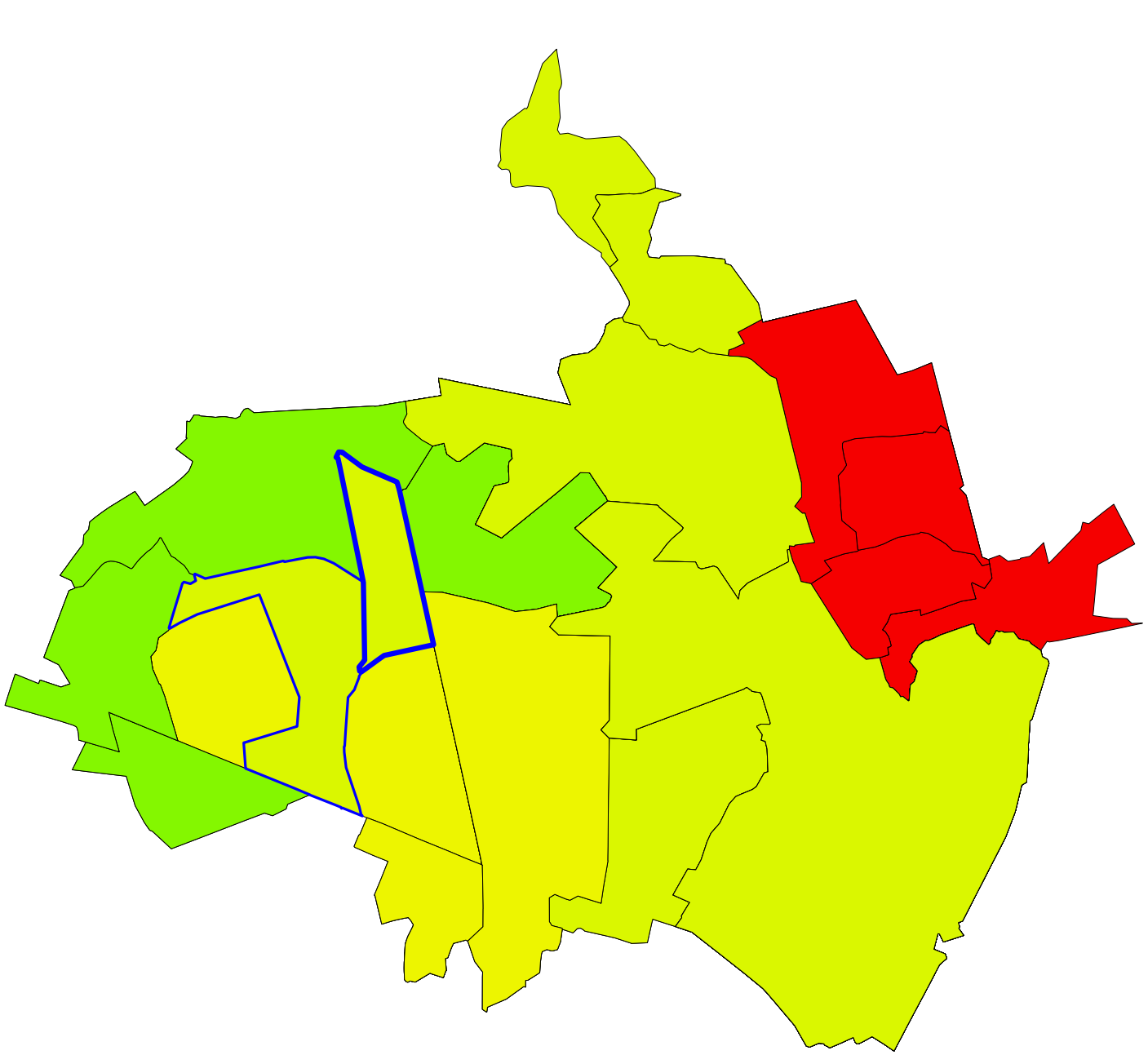


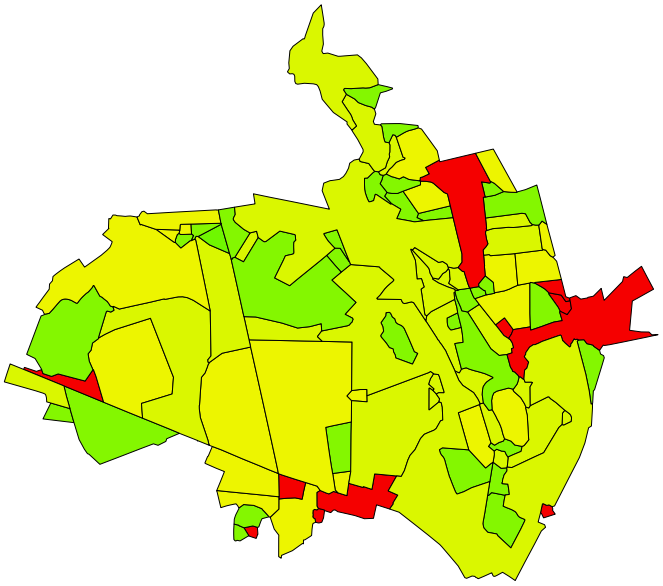


source: 92 patches

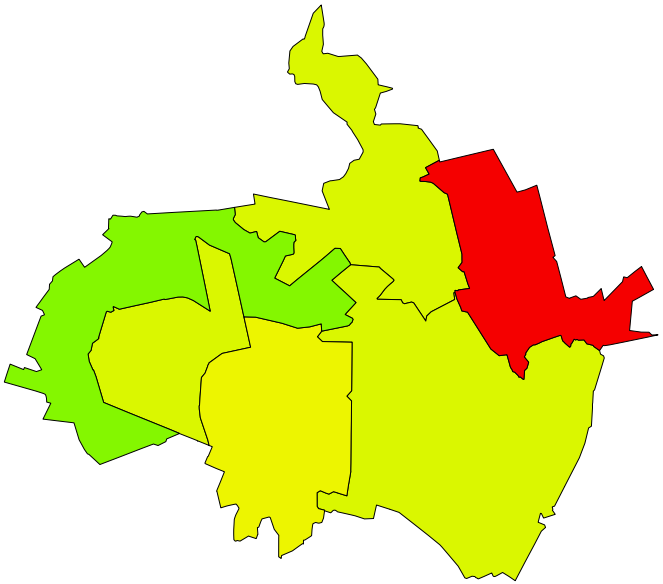


target: 6 patches

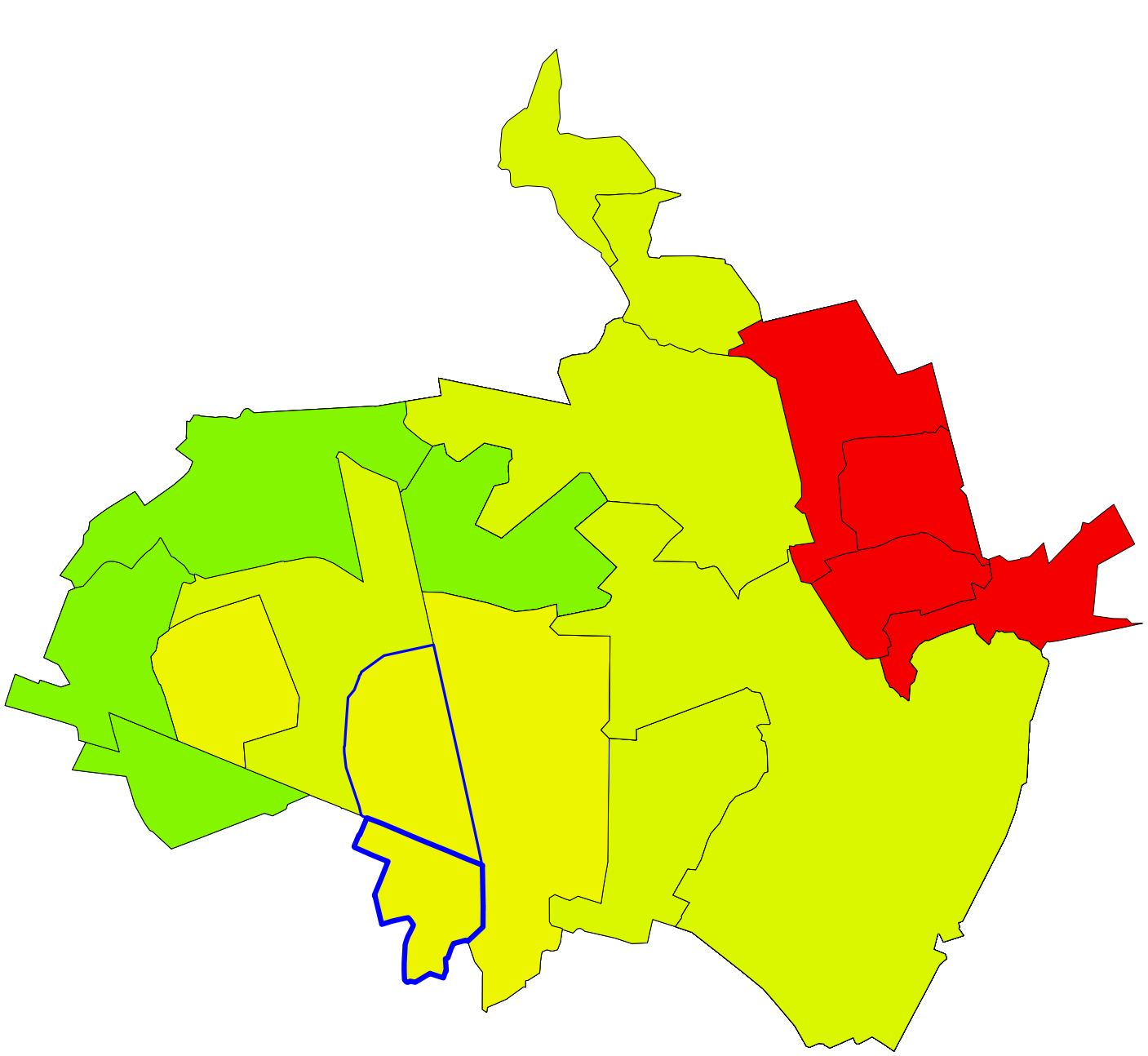


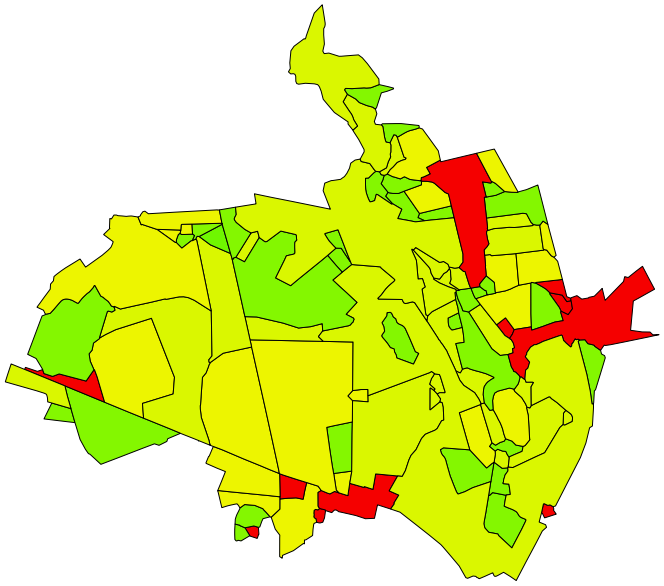


source: 92 patches

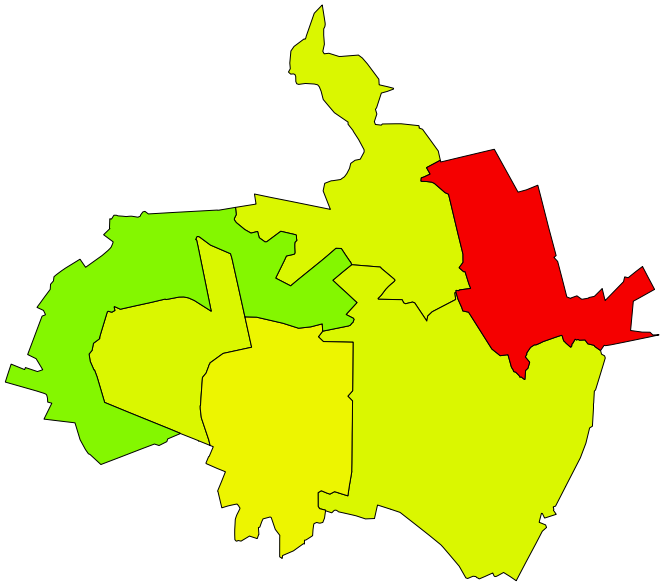


target: 6 patches

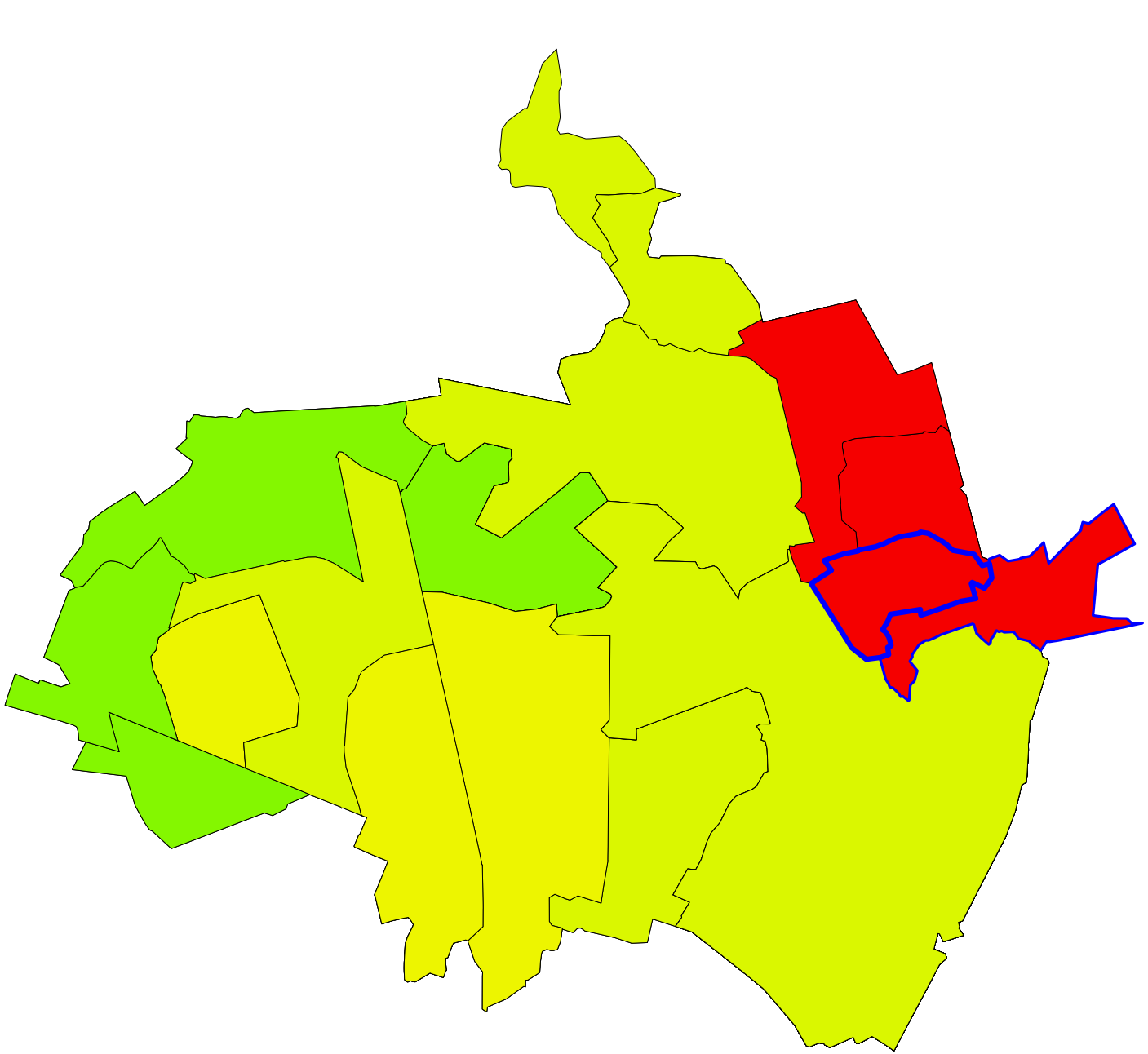


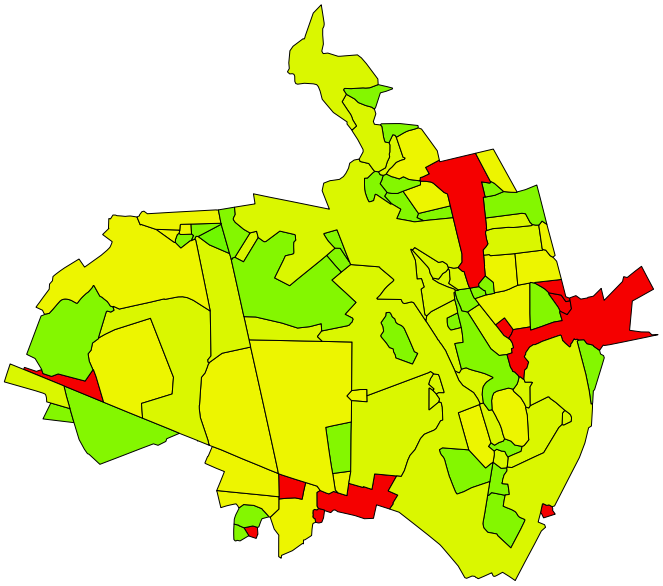


source: 92 patches

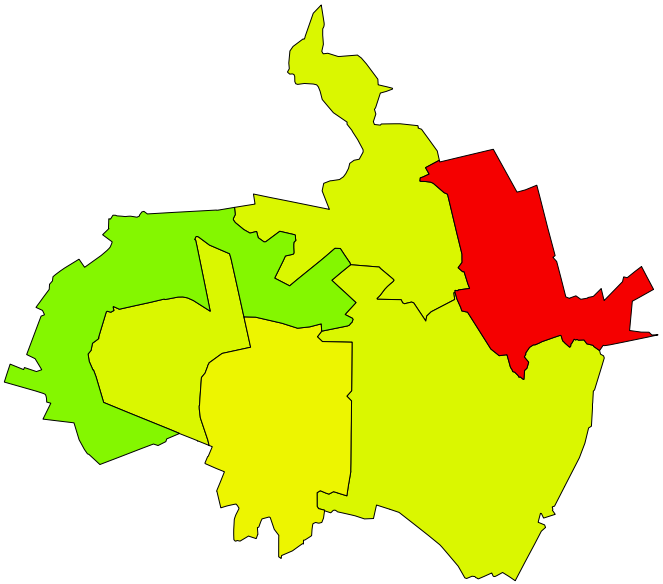


target: 6 patches

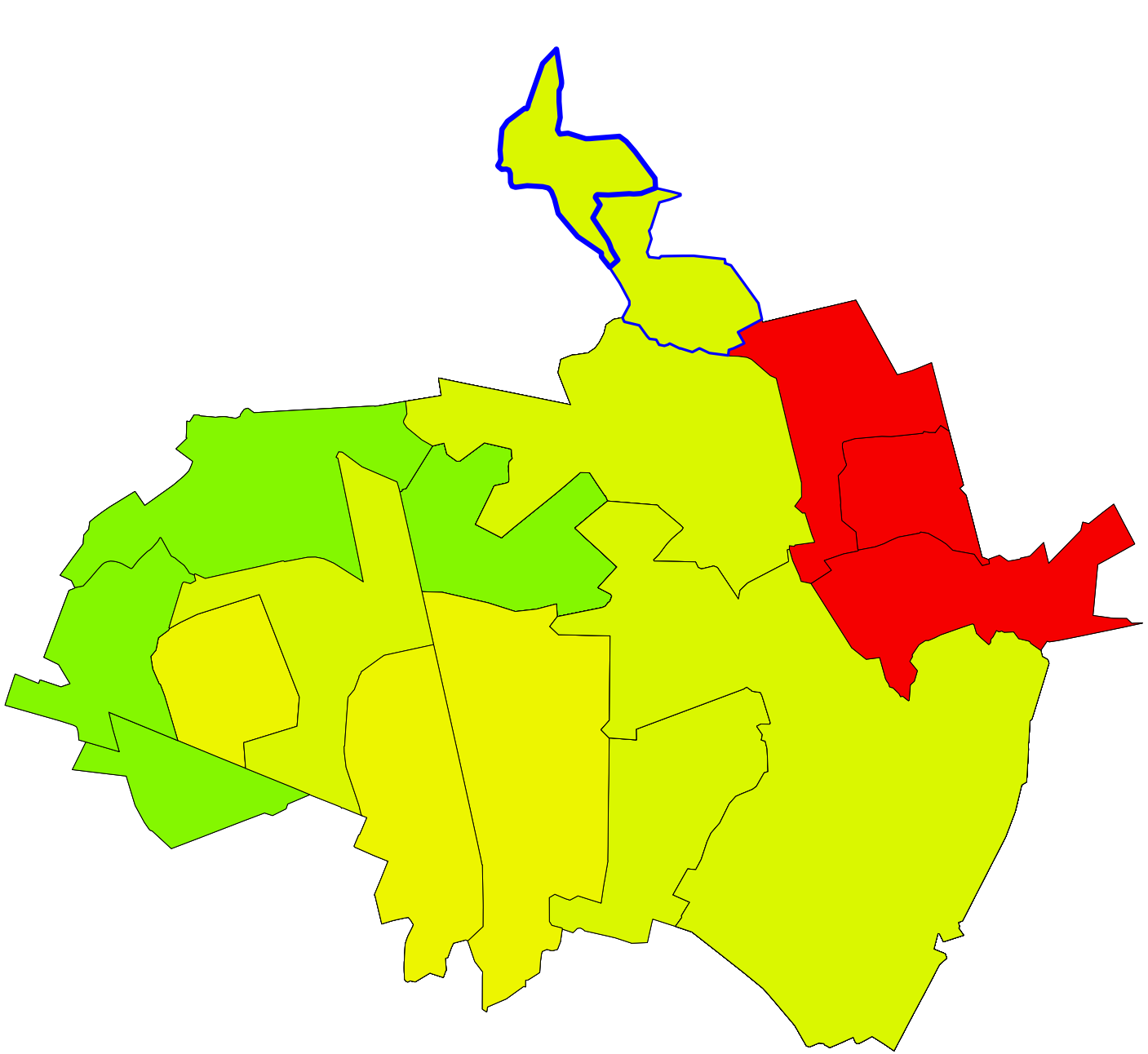


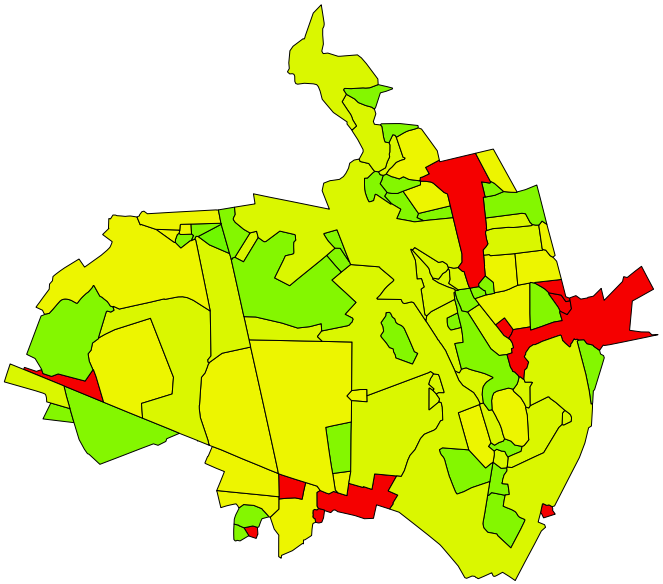


source: 92 patches

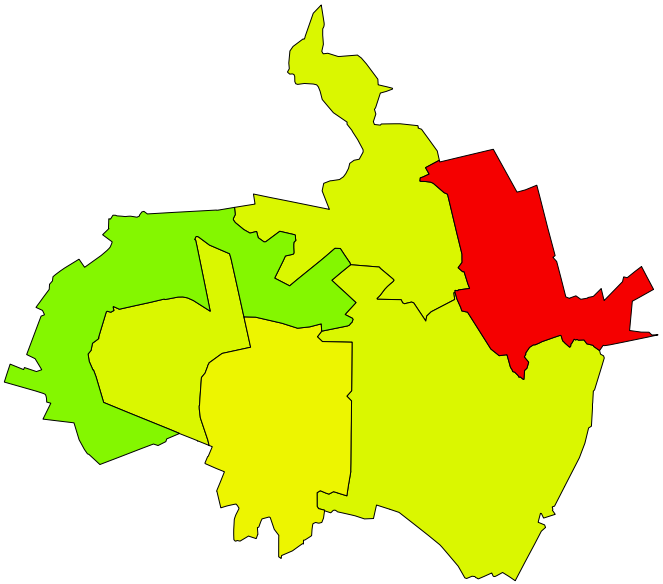


target: 6 patches

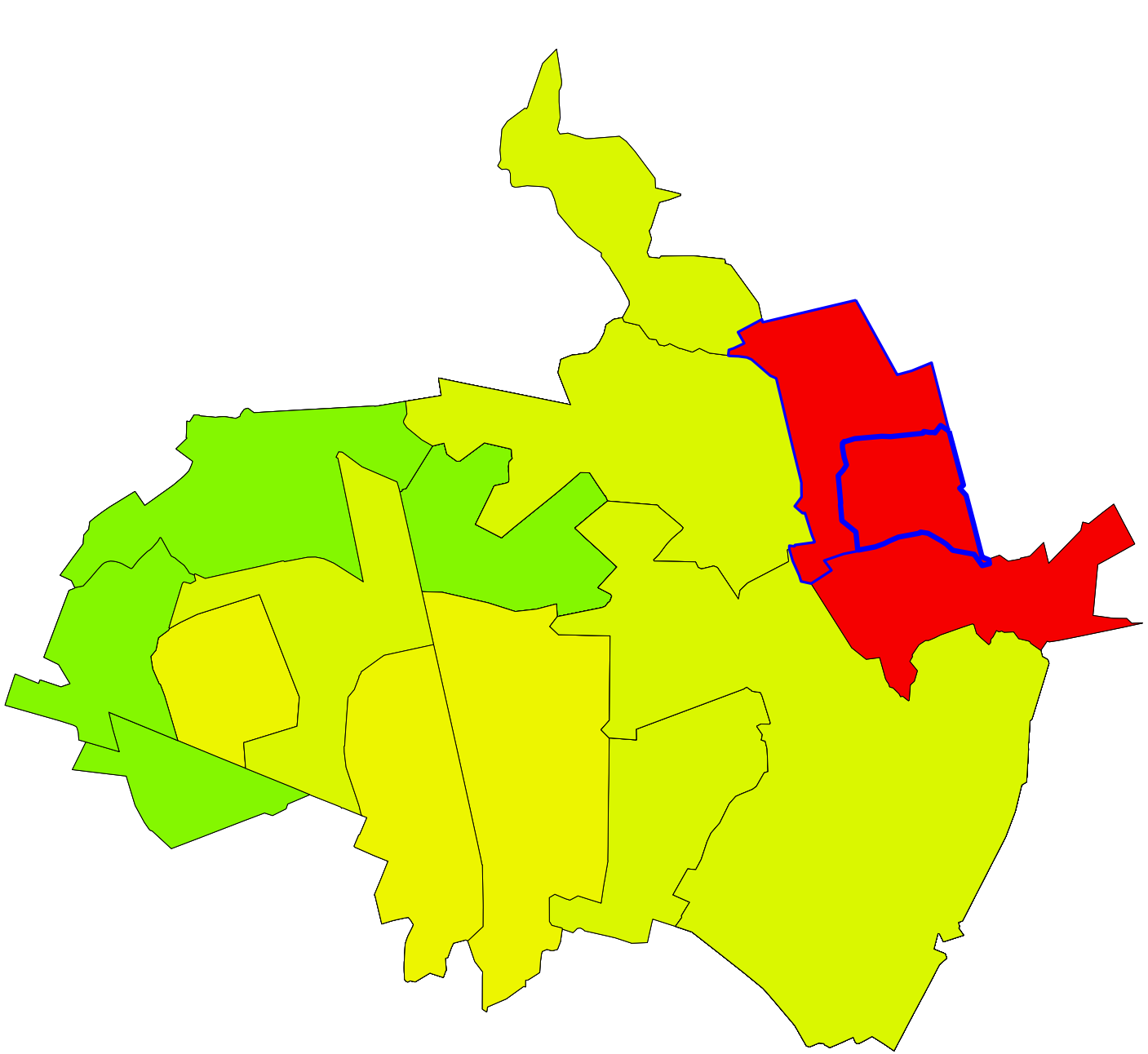


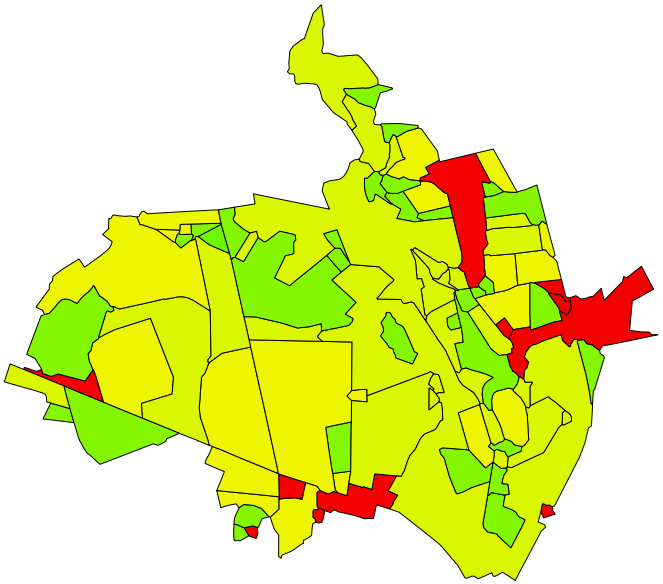


source: 92 patches

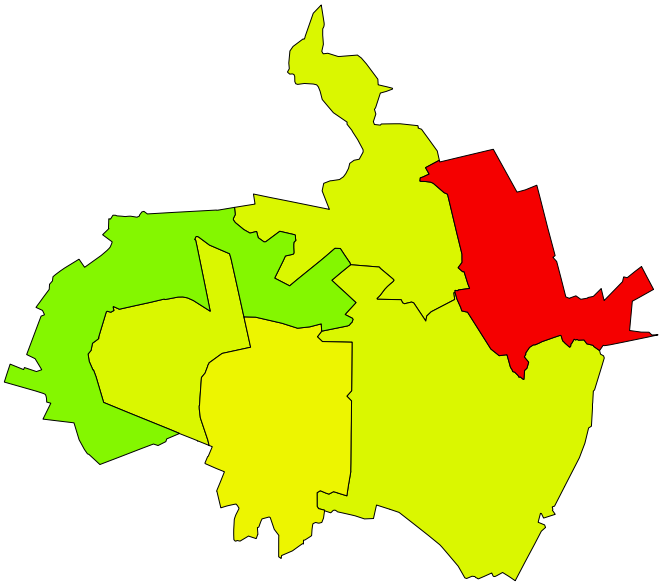


target: 6 patches

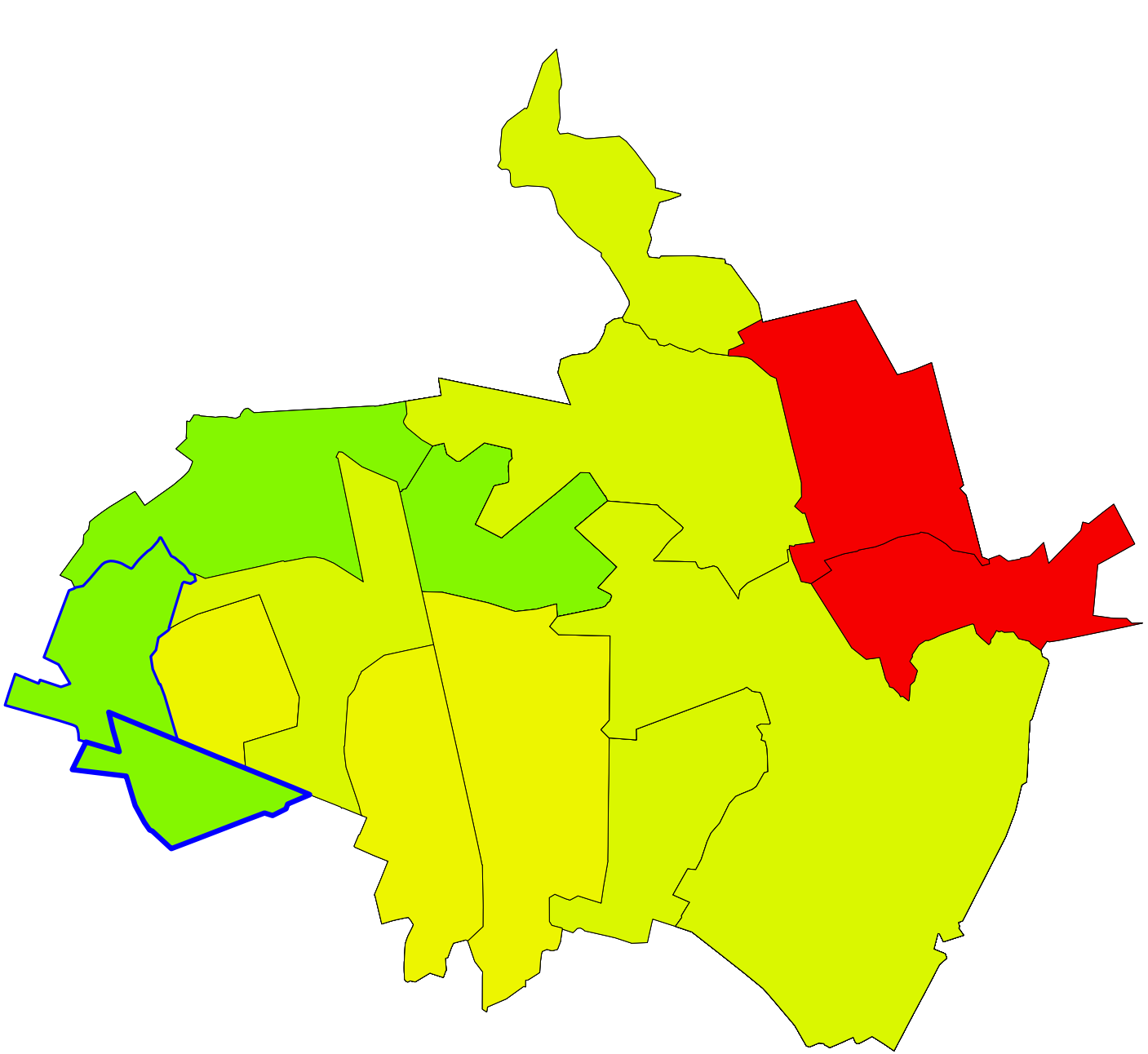


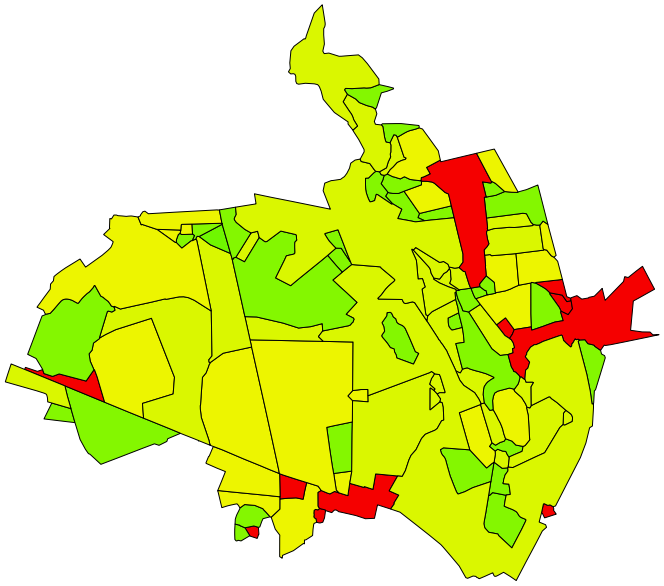


source: 92 patches

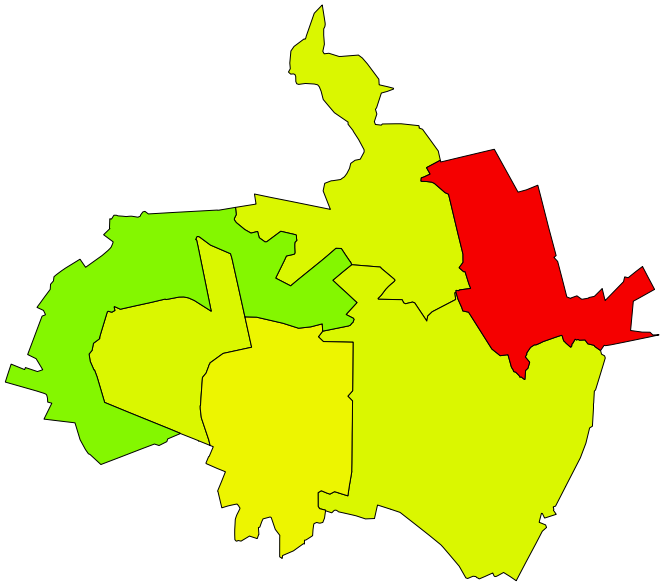


target: 6 patches

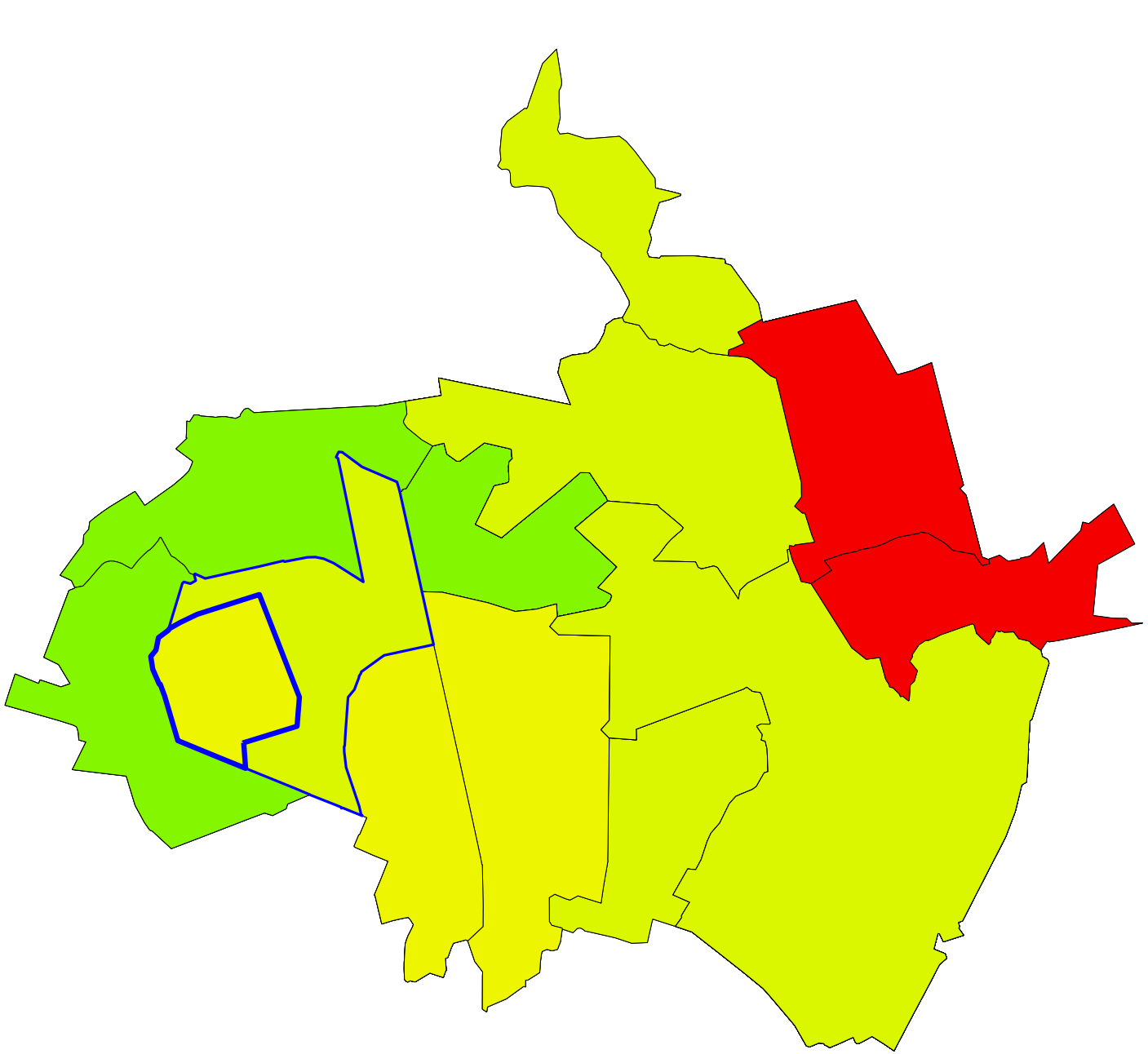


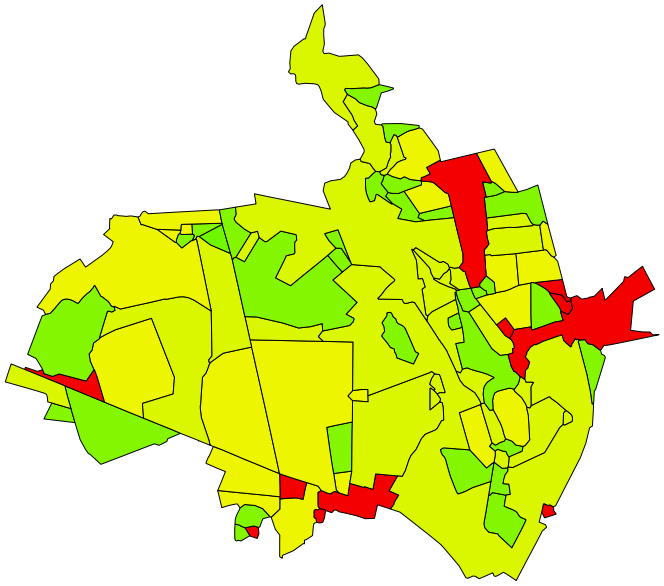


source: 92 patches

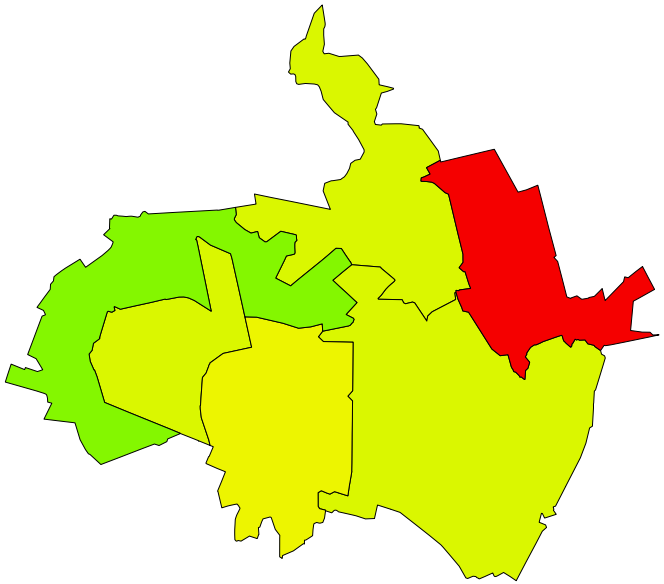


target: 6 patches

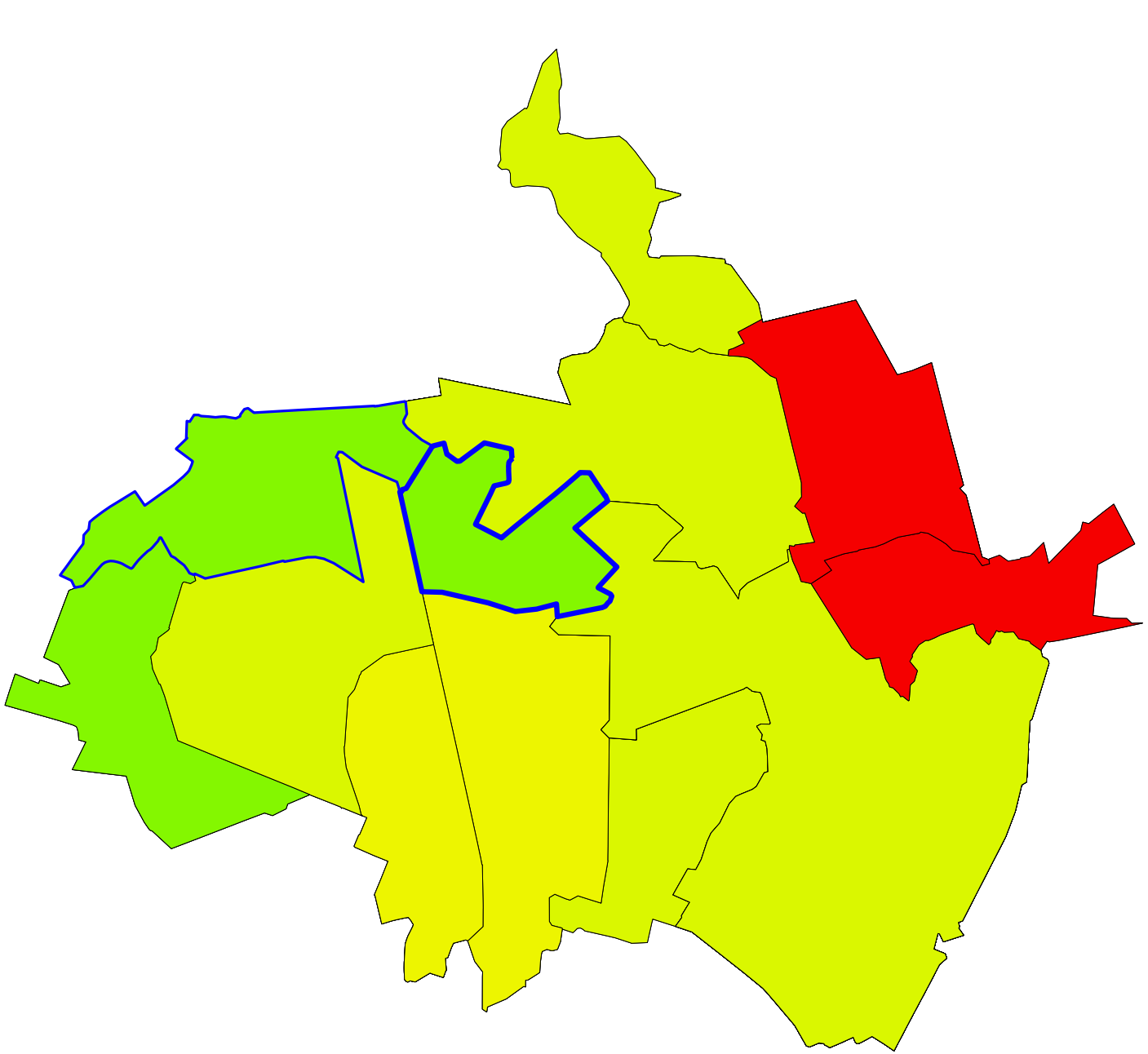


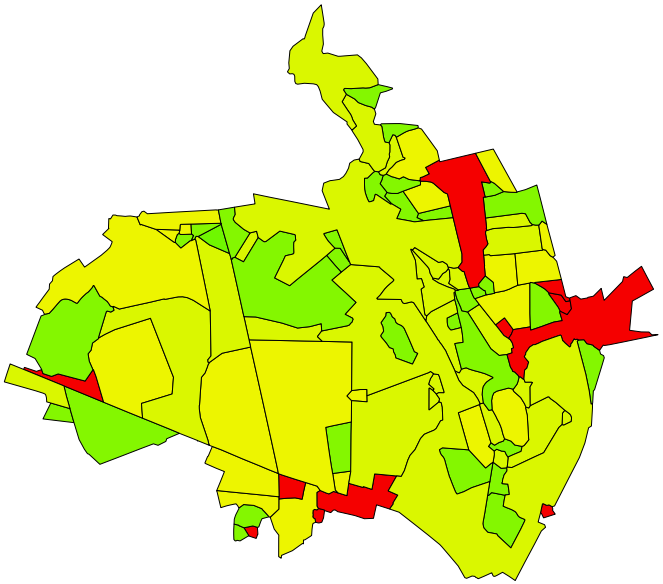


source: 92 patches

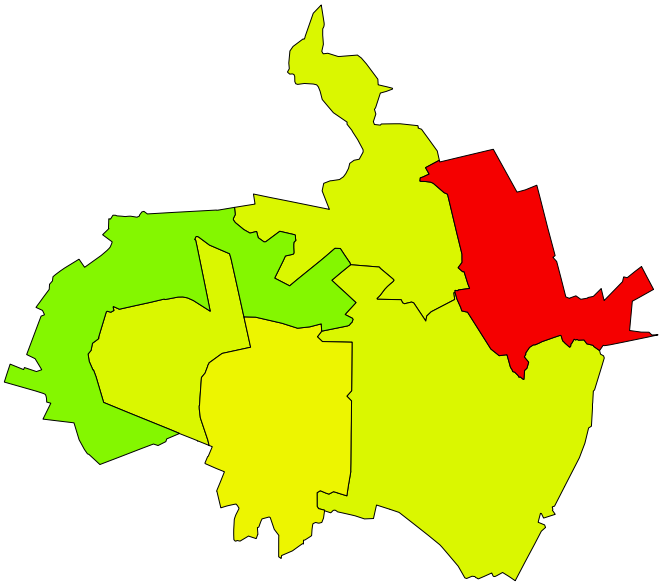


target: 6 patches

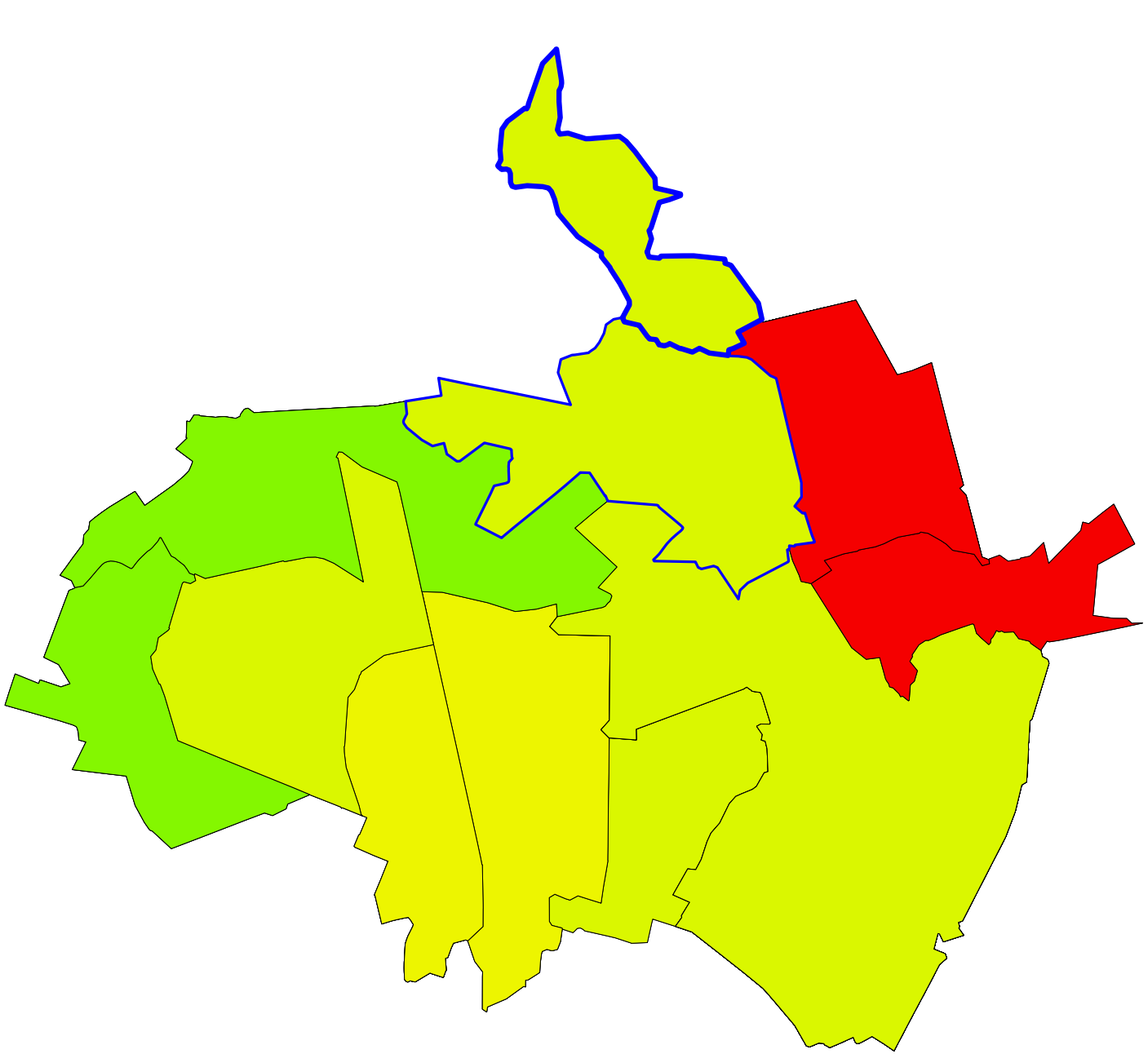


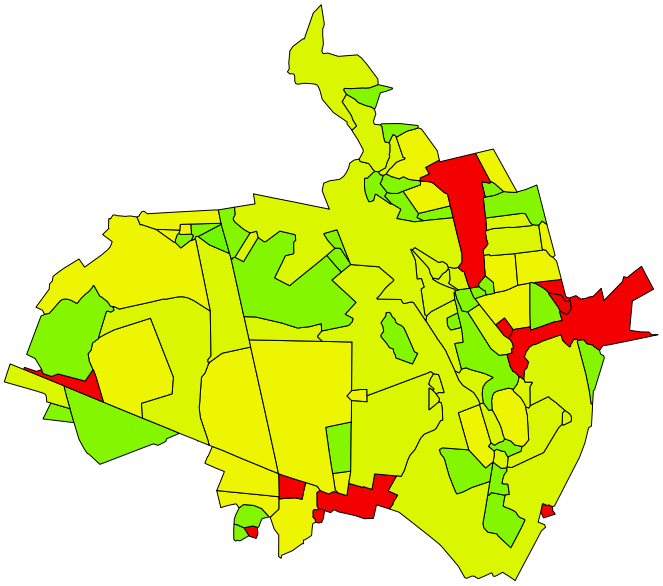


source: 92 patches

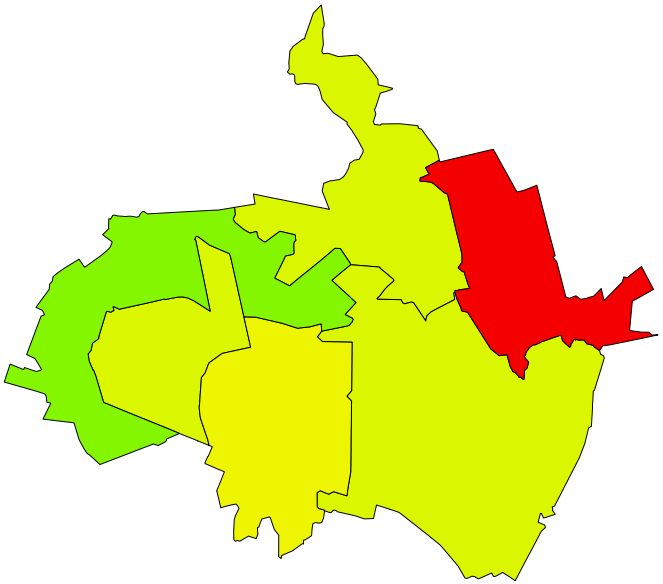


target: 6 patches

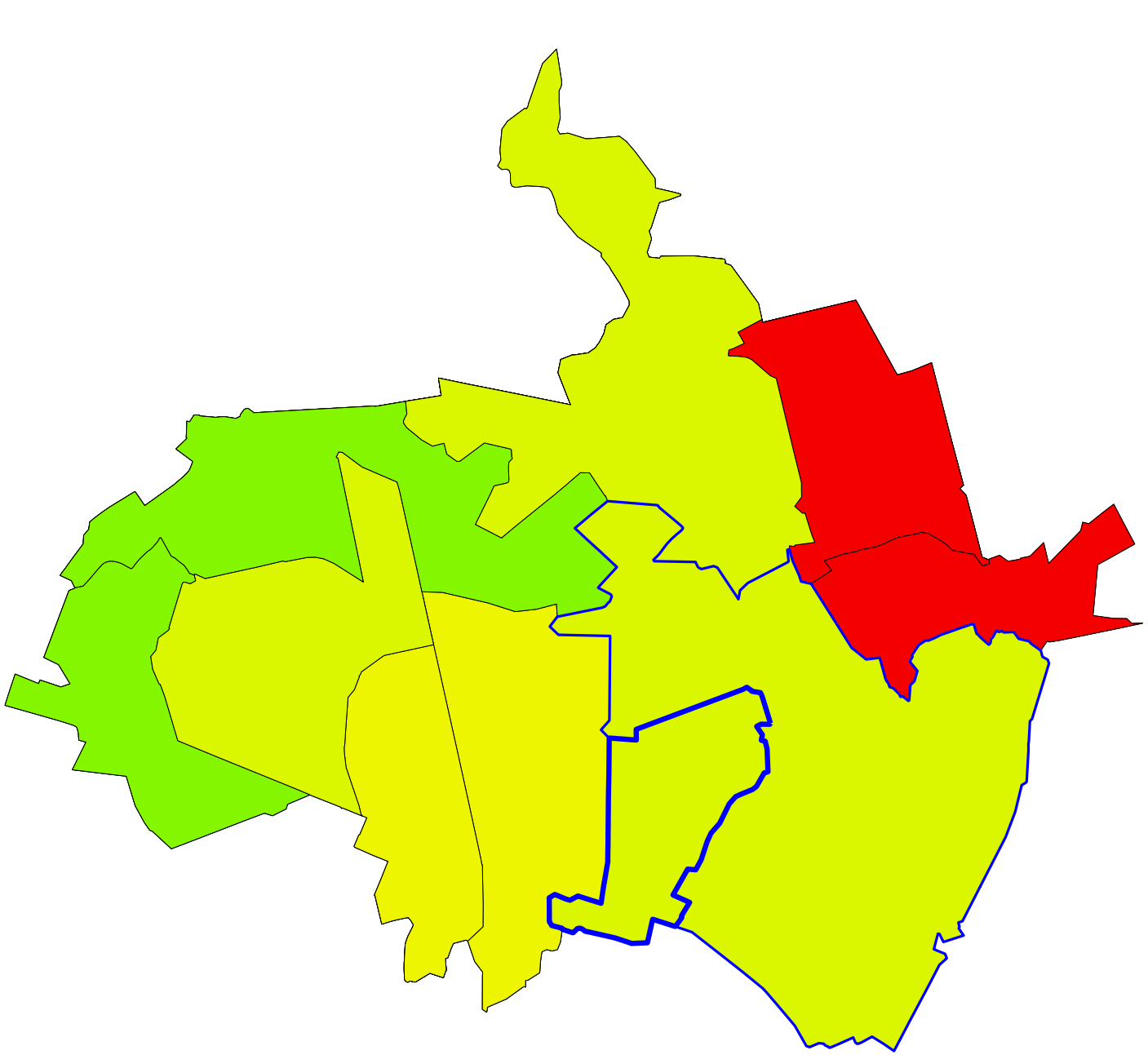


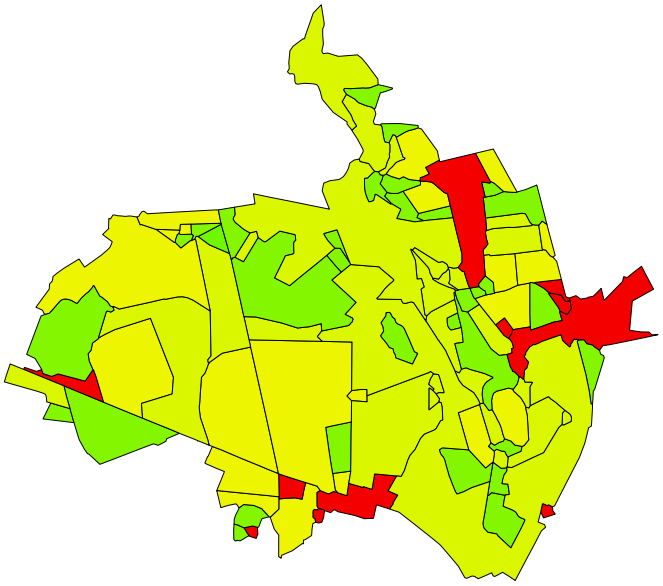


source: 92 patches

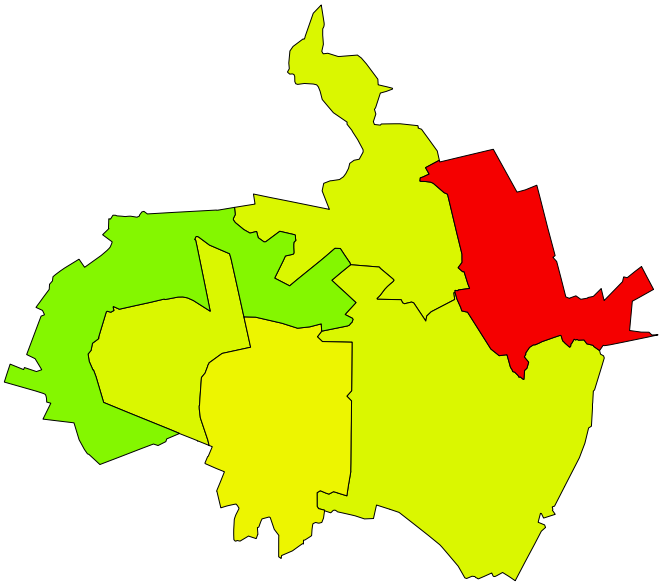


target: 6 patches

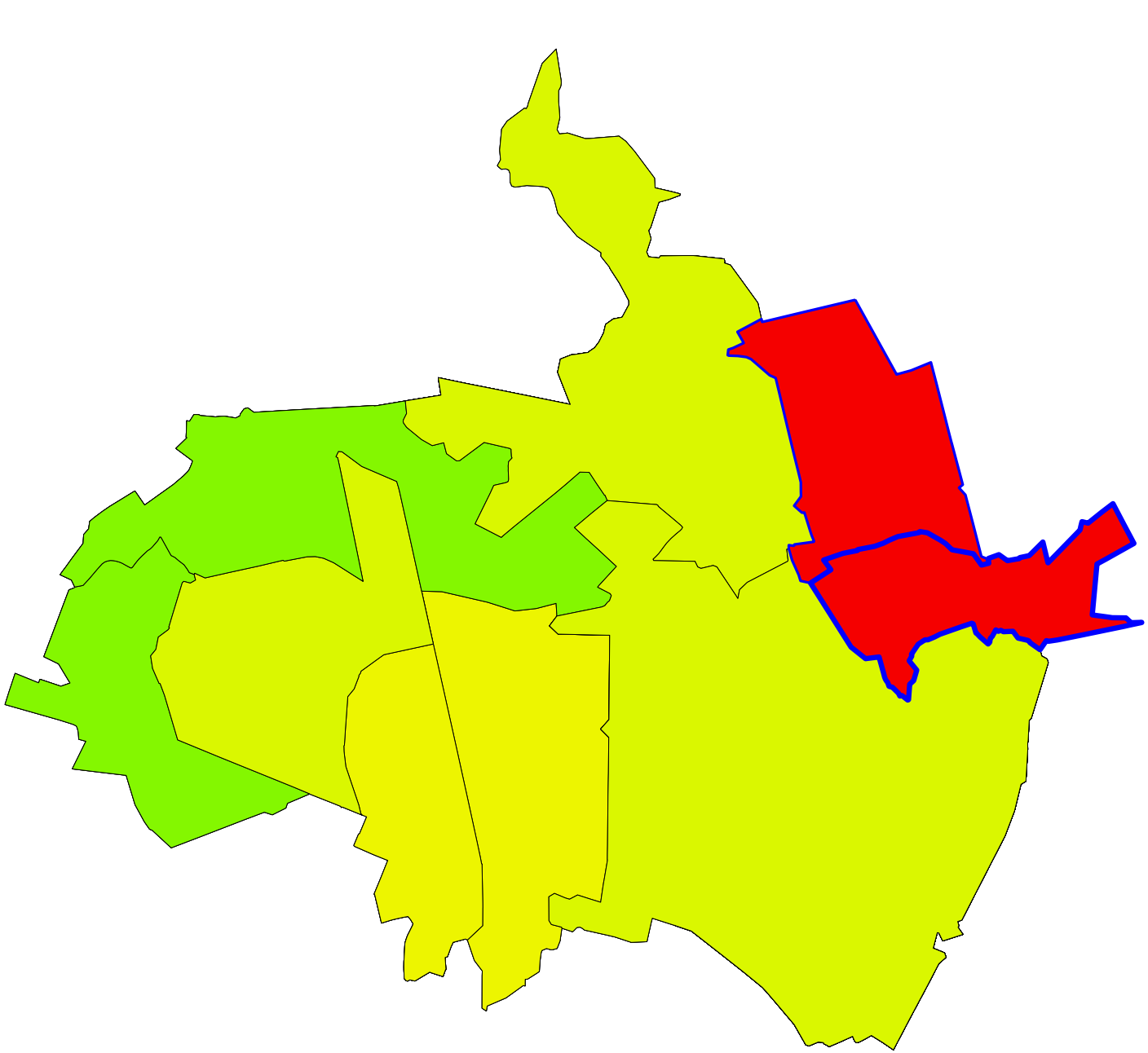


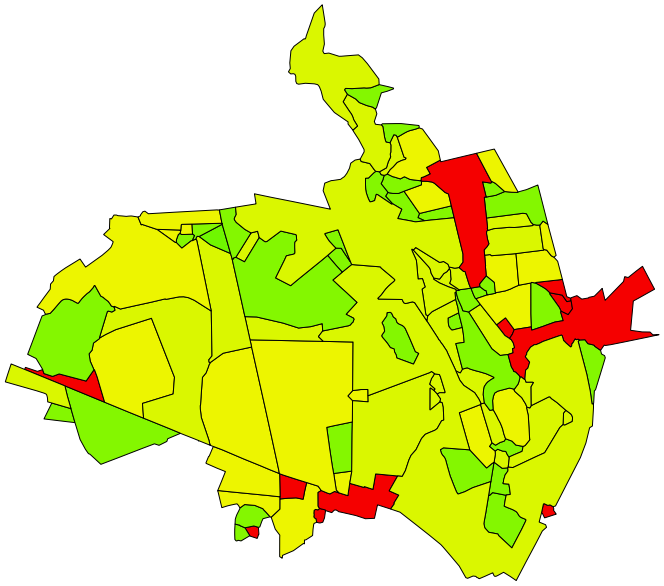


source: 92 patches

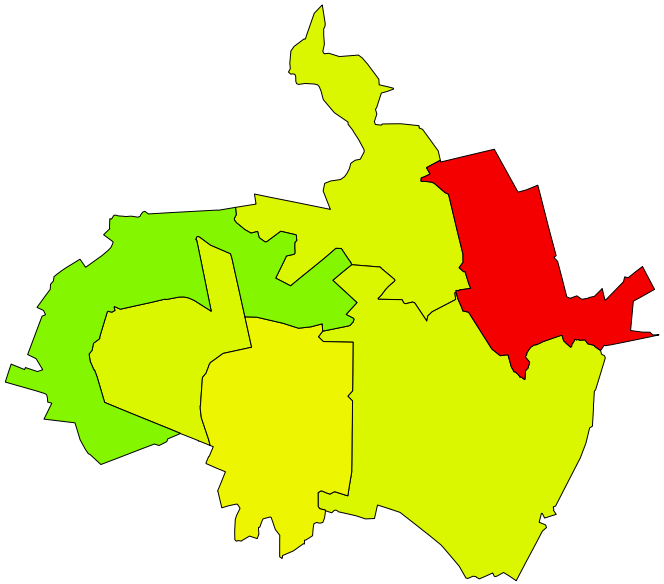


target: 6 patches

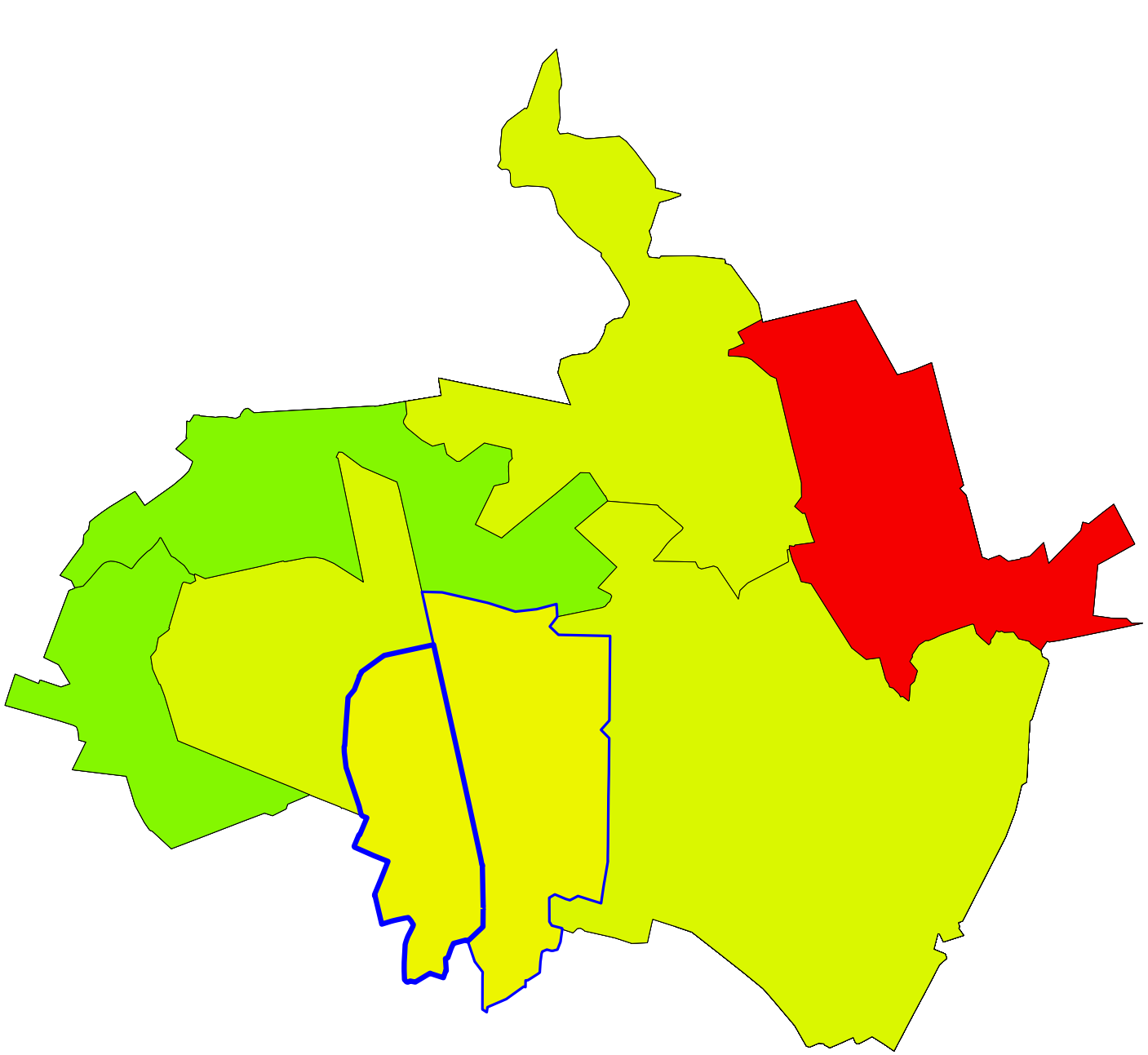


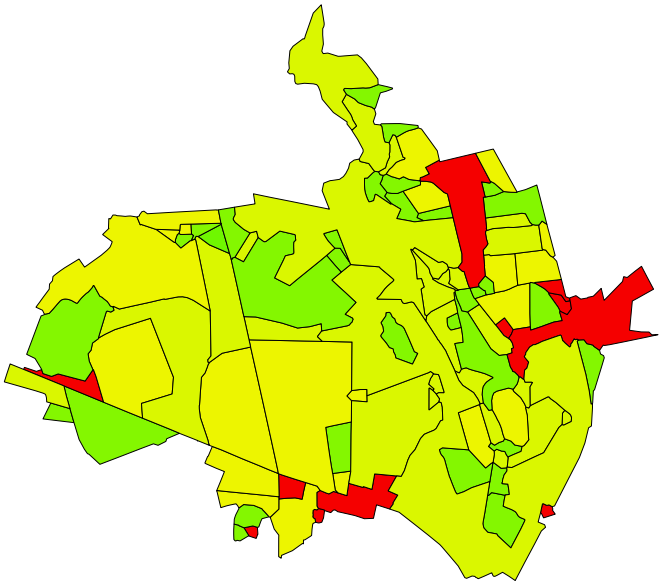


source: 92 patches

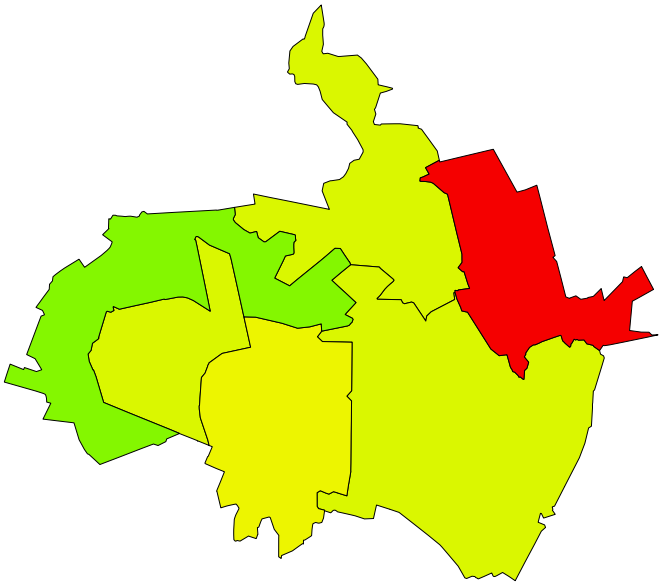


target: 6 patches

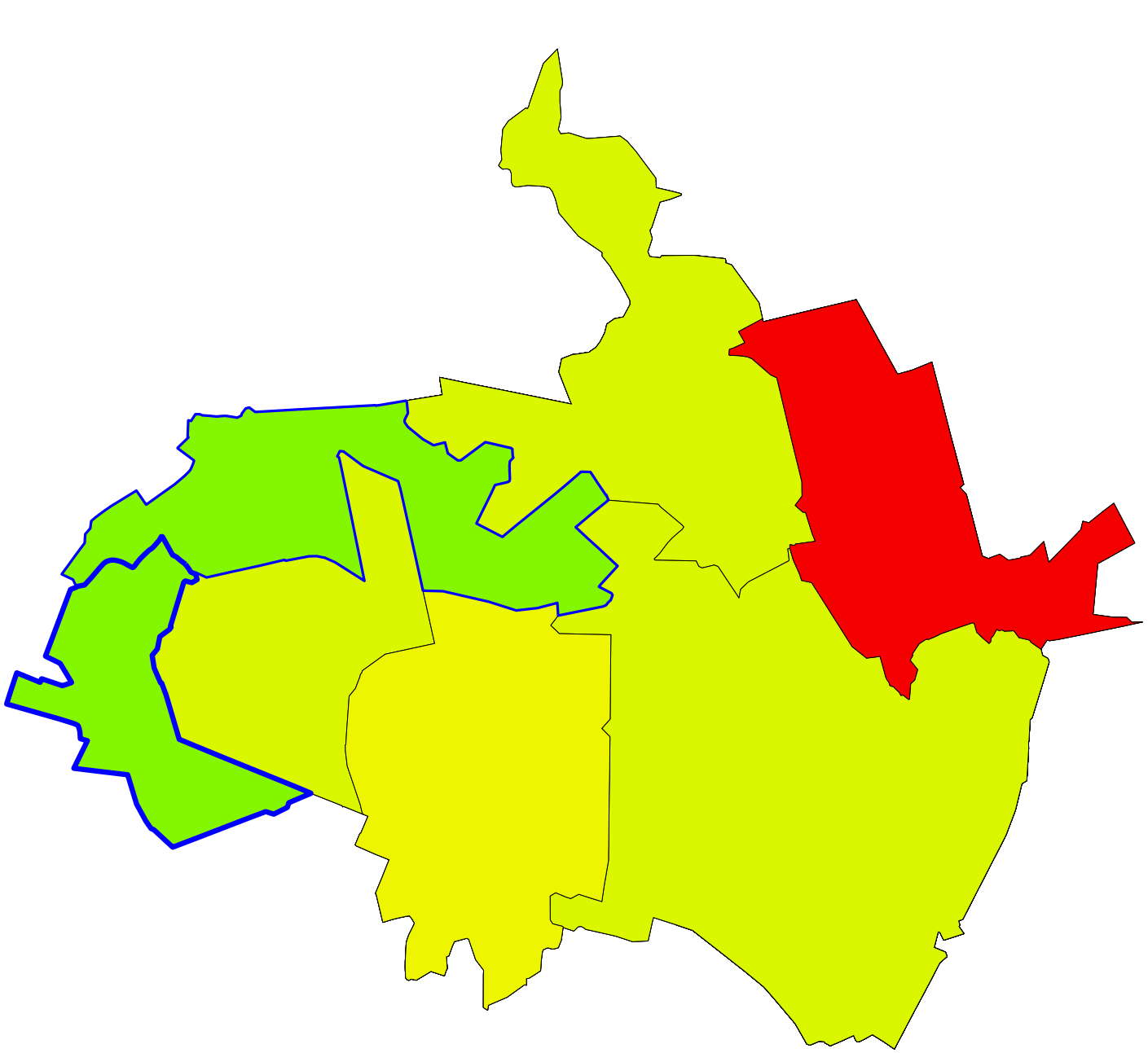


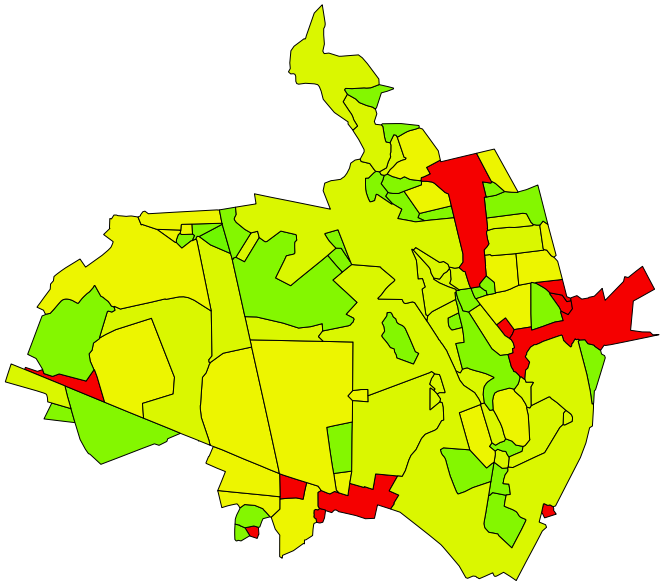


source: 92 patches

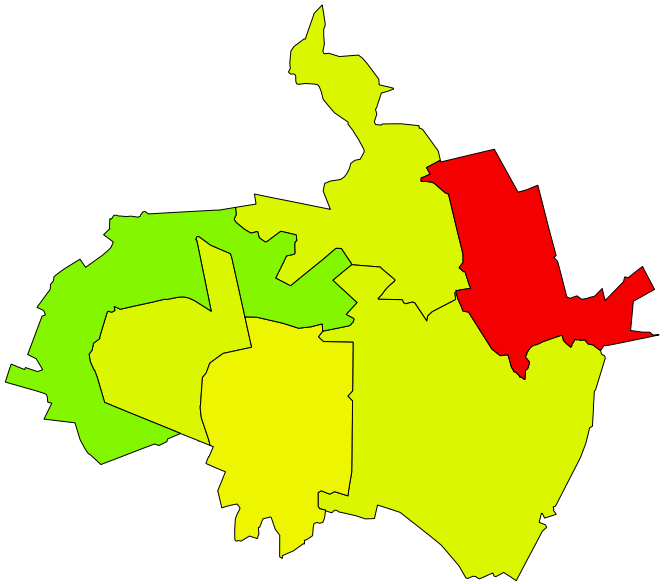


target: 6 patches

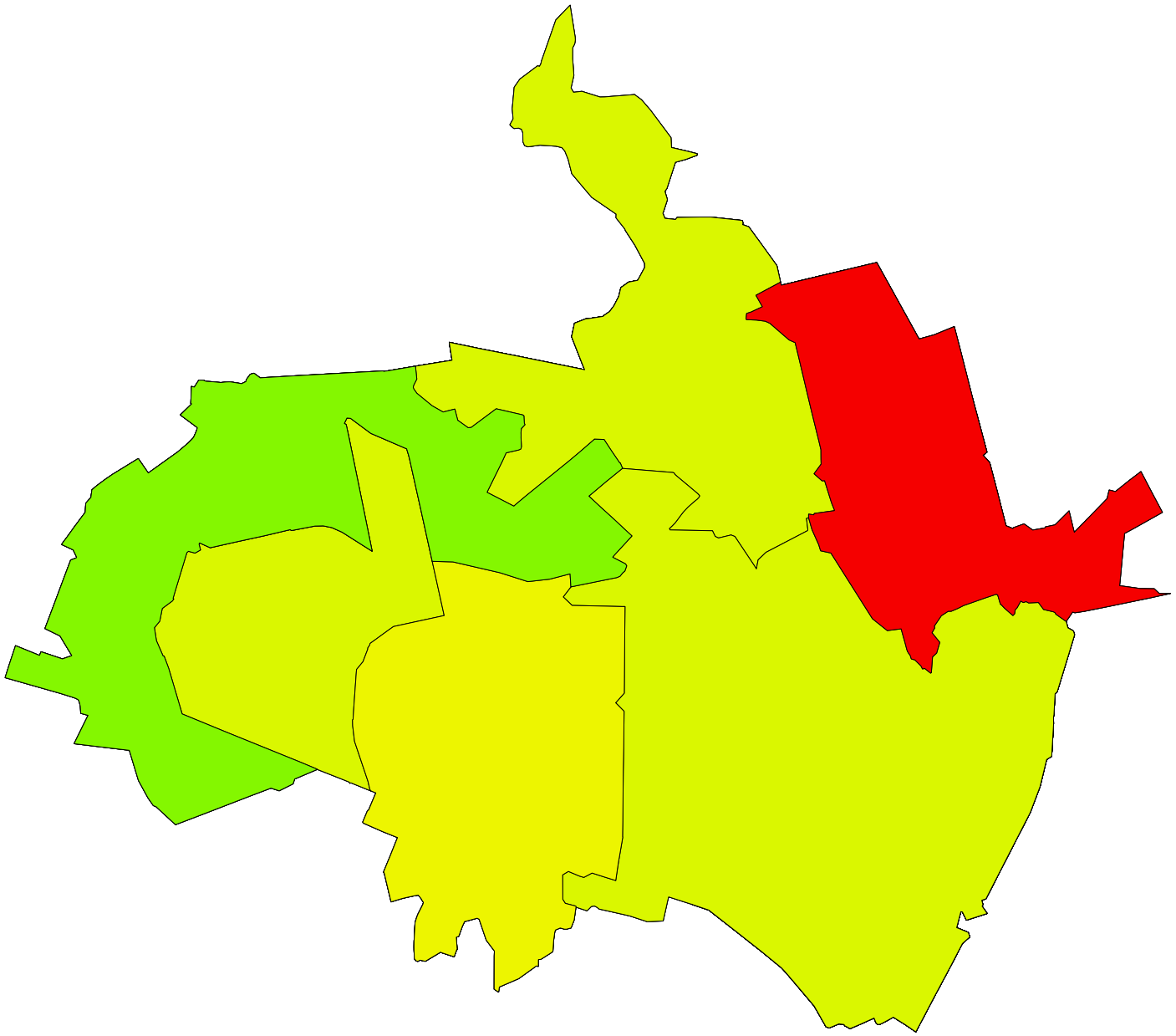


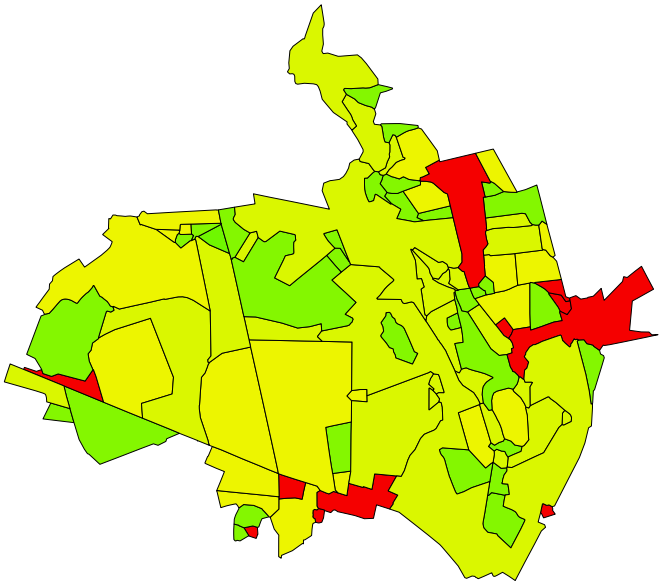


source: 92 patches

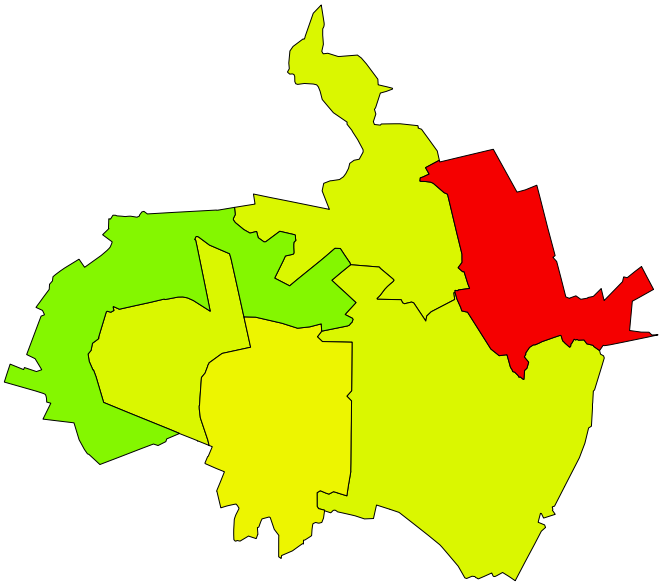


target: 6 patches





source: 92 patches



target: 6 patches

Outline

- Our Example Problem
- Methodology
- Case Study
- Concluding Remarks

Concluding Remarks

- Tried integer linear programming (ILP), A* solved more instances to optimality than ILP

Concluding Remarks

- Tried integer linear programming (ILP), A* solved more instances to optimality than ILP
- Look for better estimations: faster and find optimal solutions for more regions

Concluding Remarks

- Tried integer linear programming (ILP), A* solved more instances to optimality than ILP
- Look for better estimations: faster and find optimal solutions for more regions
- Have not considered keeping important land-cover areas for a longer time

Concluding Remarks

- Tried integer linear programming (ILP), A* solved more instances to optimality than ILP
- Look for better estimations: faster and find optimal solutions for more regions
- Have not considered keeping important land-cover areas for a longer time
- Have not considered aggregating two patches of different types into a patch of a higher-level type
  (e.g., river + lake $\Rightarrow$ waters)

Concluding Remarks

- Tried integer linear programming (ILP), A* solved more instances to optimality than ILP
- Look for better estimations: faster and find optimal solutions for more regions
- Have not considered keeping important land-cover areas for a longer time
- Have not considered aggregating two patches of different types into a patch of a higher-level type
  (e.g., river + lake $\Rightarrow$ waters)

Thank you!