# Spatially Structured Evolutionary Algorithms: Graph Degree, Population Size and Convergence Speed

Carlos M. Fernandes[1,2]

J.L.J. Laredo[3]

Agostinho C. Rosa[1]

[1]LaSEEB-ISR-IST, University of Lisbon, Portugal

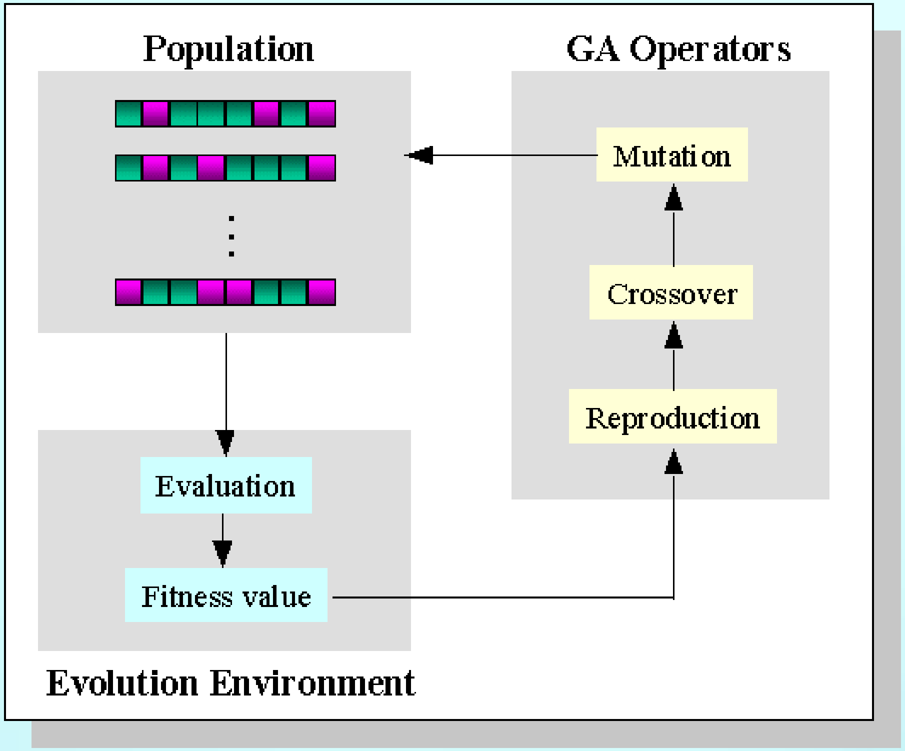
[2]Department of Computer Architecture, University of Granada, Granada, Spain

[3]LITIS, University of Le Havre, Le Havre, France

**11th International Symposium on Intelligent Distributed Computing**
**Belgrade, Serbia, October 2017**

## (panmictic) EA:

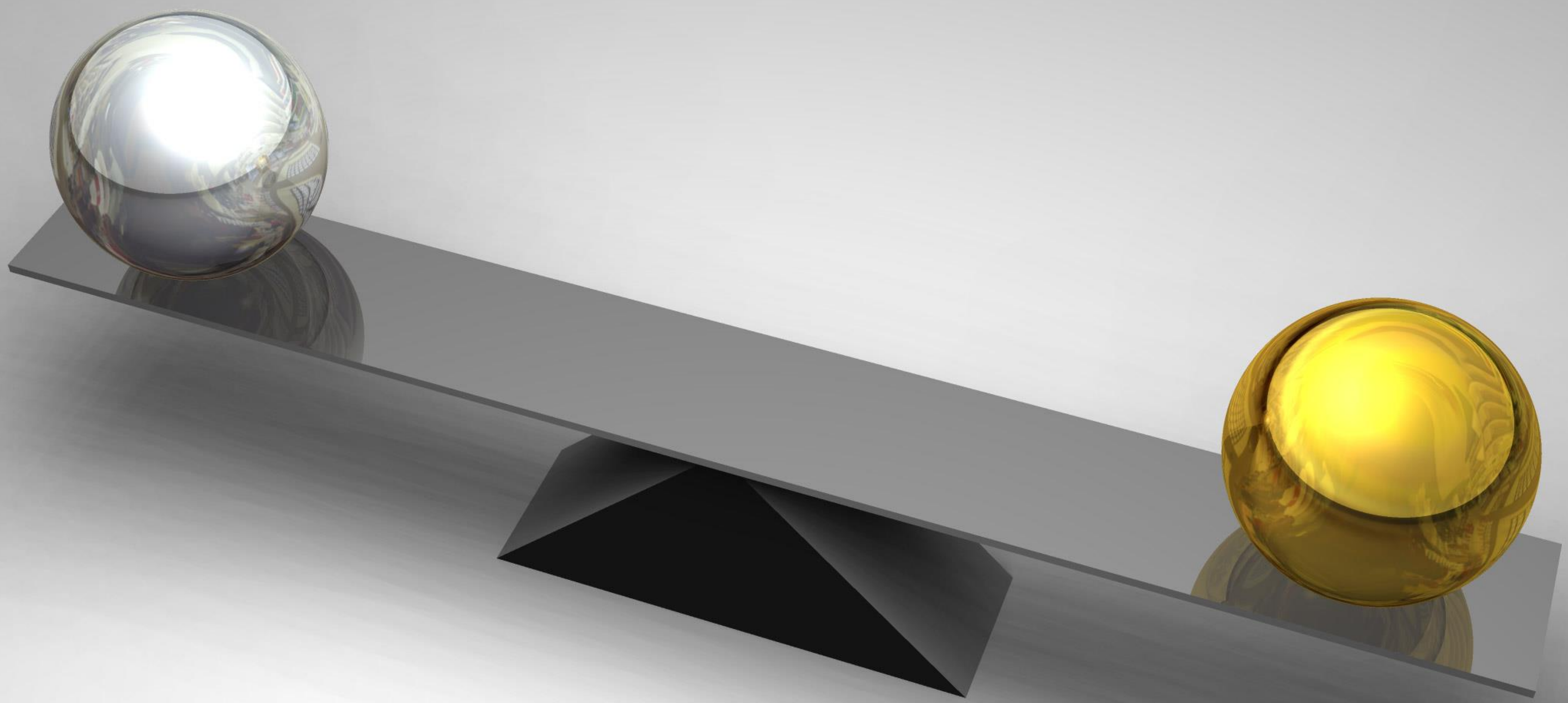
population of solutions that recombine and mutate to generate new solutions.




## spatially structured EA:

recombination is restricted to neighborhoods within the network structure.

# Objective

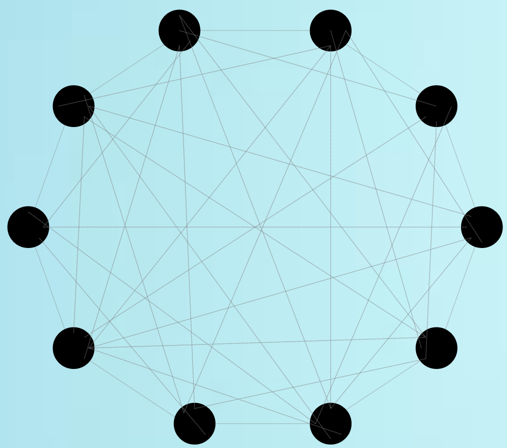


Study the relationship between population structure, population size and performance.

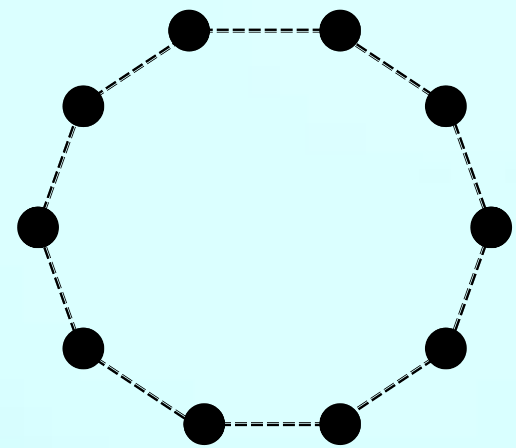
# Why?

- EA performance is strongly dependent on population size.
- Topology and optimal population size.
- Different conclusions?

# Population Structure

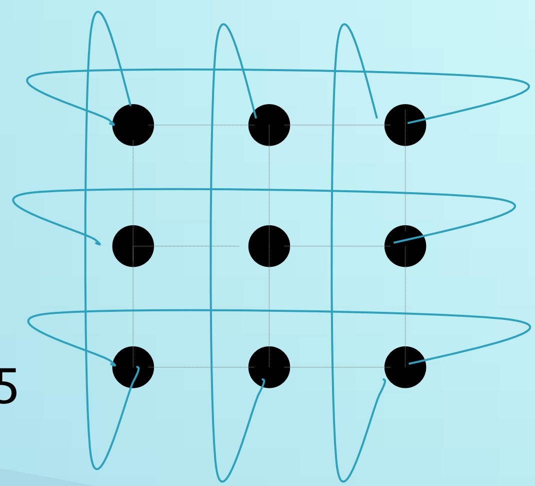


$k = n - 1$ (panmictic population)



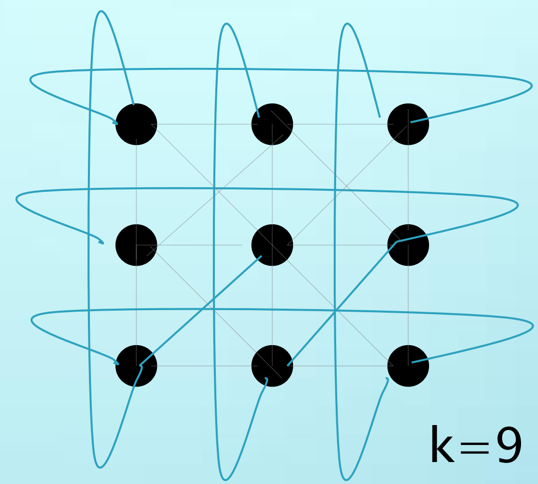
$k = 2$

*von Neumann*



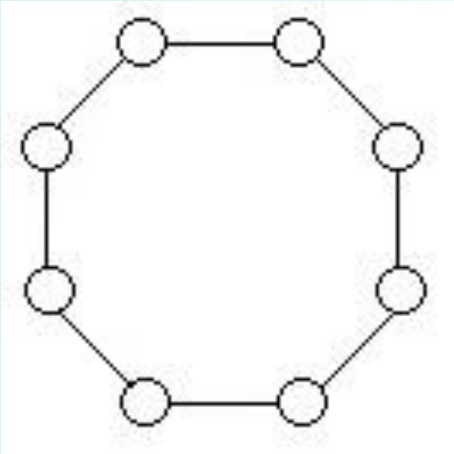
k=5

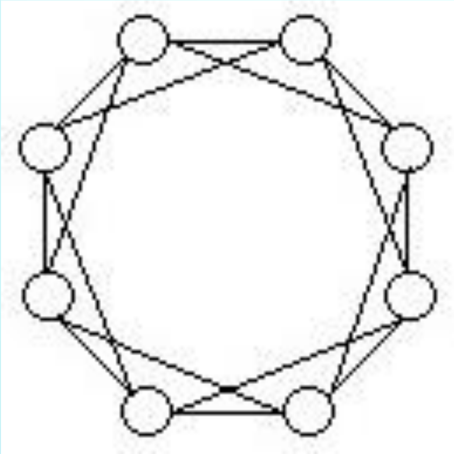
*Moore*

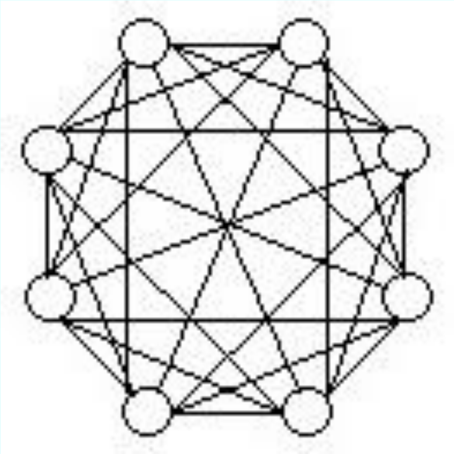


k=9

# Population Structure







$k = 2$ $k = 4$ $k = 7 = n - 1$

Regular graphs for population size $n = 8$.

# Experimental Setup

- Bi-section method to determine optimal population size.

**Algorithm 2: Bisection method**

1. Start with small n
2. Double n until EA convergence criteria is met
3. (min,max)=(n/2,n)
4. repeat until (max-min)/min < T

    n =(min+max)/2

    if n leads to convergence criteria

      then max = n    else min = n
5. Compute the statistics for the problem size using population size = max

# Experimental Setup

- Onemax, 2-trap, 3-trap, 4-trap, MMDP

  deceptive

- Synchronous cellular EA

- Binary tournament, two-point crossover, bit-flip mutation

# Results: optimal population size

| | onemax | 2-trap | 3-trap | 4-trap | MMDP |
|---|---|---|---|---|---|
| $k = 2$ | 400 | 500 | 350 | 400 | 500 |
| $k = 4$ | 450 | 500 | 350 | 450 | 500 |
| $k = 8$ | 500 | 550 | 350 | 500 | 500 |
| $k = 16$ | 550 | 700 | 400 | 550 | 500 |
| $k = 32$ | 800 | 750 | 500 | 550 | 550 |
| $k = 64$ | 1000 | 1000 | 650 | 650 | 650 |
| $k = 128$ | 1200 | 1300 | 700 | 750 | 750 |
| $k = n - 1$ | 2200 | 2275 | 1100 | 1200 | 800 |

# Results

| | onemax | 2-trap | 3-trap | 4-trap | MMDP |
|---|---|---|---|---|---|
| $k = 2$ | 319,986.21 | 348,483.33 | 133,712.07 | 146,560.00 | 168,266.67 |
| | ±17,253.56 | ±22,773.05 | ±16,085.69 | ±18,423.25 | ±29,590.81 |
| $k = 4$ | 219,930.00 | 222,433.33 | 86,205.00 | 104,167.24 | 108,866.67 |
| | ±12,784.26 | ±26,610.77 | ±10,812.42 | ±12,786.66 | ±14,151.84 |
| $k = 8$ | 157,233.33 | 147,,836.21 | 54,961.67 | 73,206.90 | 66,100.00 |
| | ±7,747.89 | ±1,1261.46 | ±5,178.70 | ±10,743.37 | ±8,515.10 |
| $k = 16$ | **114,210.34** | 129,173.33 | 41,701.67 | 51,645.00 | 44,724.14 |
| | **±5,271.88** | ±7,495.93 | ±4,602.52 | ±4,915.85 | ±5,630.89 |
| $k = 32$ | 122,560.00 | **99,795.00** | 38,683.33 | 40,425.00 | 35,806.90 |
| | ±5,366.15 | **±5,847.96** | ±3,100.29 | ±4,175.58 | ±2,988.51 |
| $k = 64$ | 117,933.33 | 104,068.97 | 40,913.33 | **37,812.07** | 33,979.31 |
| | ±4,448.42 | ±4,008.30 | ±2,966.80 | **±2,395.00** | ±3,254.16 |
| $k = 128$ | 120,331.03 | 110,196.67 | **37,727.59** | 38,700.00 | 33,725.00 |
| | ±3,887.07 | ±4,601.69 | **±1,947.32** | ±3,275.69 | ±2,904.63 |
| $k = n - 1$ | 181,462.07 | 160,463.33 | 51,920.00 | 49,646.67 | **33,296.55** |
| | ±5,010.59 | ±3,995.89 | ±3,792.95 | ±3,989.5 | **±2,142.51** |

# Results

| | onemax | 2-trap | 3-trap | 4-trap | MMDP |
|---|---|---|---|---|---|
| $k = 2$ | 95,900.00 | 235,716.67 | 99,497.22 | 110,708.33 | 124,673.91 |
| | ±3,626.24 | ±23,382.20 | ±15799.90 | ±21,653.64 | ±58,068.80 |
| $k = 4$ | 72,460.00 | 148,950.00 | 65,856.00 | 67,452.17 | 151,840.91 |
| | ±2,769.36 | ±16,527.33 | ±12,553.7 | ±10,967.43 | ±200,212.74 |
| $k = 8$ | 55,493.33 | 117,900.00 | 58,403.95 | 43,233.33 | 114,608.70 |
| | ±2,183.79 | ±49,055.90 | ±89,744.99 | ±5,718.13 | ±166,207.75 |
| $k = 16$ | 43,646.67 | 217,041.67 | 25,300.00 | 26,914.29 | 116,761.36 |
| | ±2,058.24 | ±371,359.30 | ±2,710.82 | ±4,989.13 | ±230,706.28 |
| $k = 32$ | 35,793.33 | 423,041.67 | 17,675.00 | 18,600.00 | 162,950.00 |
| | ±1,960.64 | ±203,743.01 | – | – | ±230,706.28 |
| $k = 64$ | 32,866.67 | 615,841.67 | – | 15,000.00 | 86,708.33 |
| | ±1,590.78 | ±228,150.17 | – | – | ±119,648.17 |
| $k = 128$ | 32,333.33 | 608,983.33 | – | 13800.00 | 287,468.75 |
| | ±2,074.03 | ±239,684.09 | – | – | ±304,905.86 |
| $k = n - 1$ | 31,600.00 | 611,441.67 | – | – | 127,277.78 |
| | ±1,702.74 | ±232,371.70 | – | – | ±153,606.36 |

# Conclusions

- Graphs with lower degree require smaller populations.
- However, the larger populations required by graphs with higher degree converge faster to global optima.
- When the population is properly set, higher degree or even panmictic populations are more efficient.
- Conclusions on the performance of the different structures are different if the population size is set to the same value for all configurations.

# Future Work

- Random, small-world and dynamic structures.
- Takeover times of different graphs, with different population size.
- Performance of the different structures with growing string size (scalability tests).
- Parallel implementations.


Hvala lepo!
Questions?