Hiding Points in Polygons using Approximation Algorithms

Antonio L. Bajuelos

Mafalda Martins

Universidade de Aveiro & CEOC

Santiago Canales

Universidad Pontificia Comillas

Gregorio Hernández

Universidad Politécnica de Madrid

Guarding

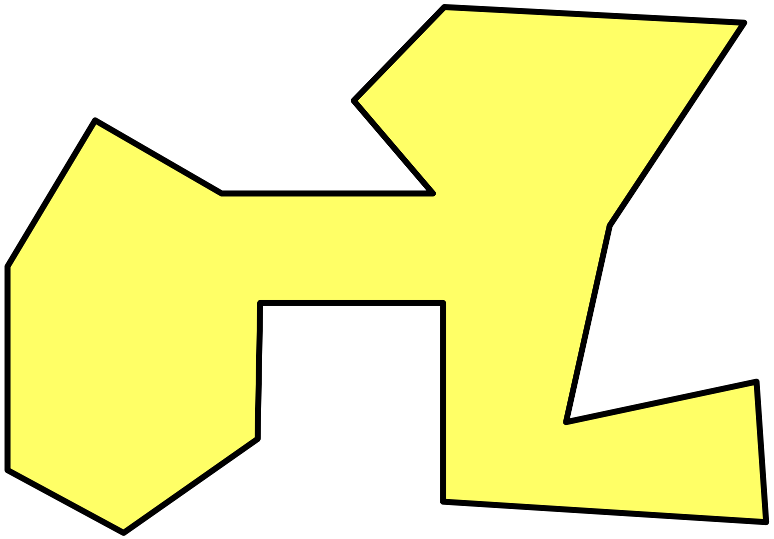
- **Visibility Problems: Guarding** and **Hiding**
- Input: simple polygon P
  - **Guarding:** find a minimum number of points (guards) in P, such that each point in P is seen by at least one guard

Guarding

- **Visibility Problems: Guarding** and **Hiding**

- Input: simple polygon P

  - **Guarding:** find a minimum number of points (guards) in P, such that each point in P is seen by at least one guard

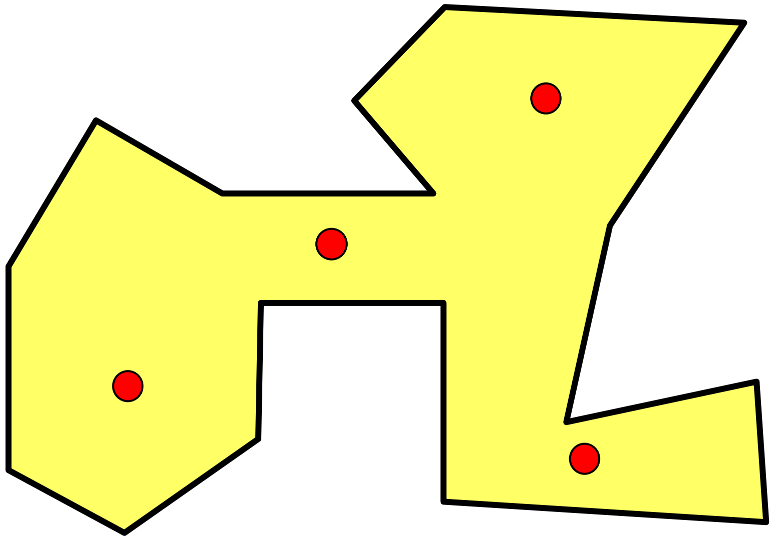


Guarding

- **Visibility Problems: Guarding** and **Hiding**

- Input: simple polygon P

  - **Guarding:** find a minimum number of points (guards) in P, such that each point in P is seen by at least one guard

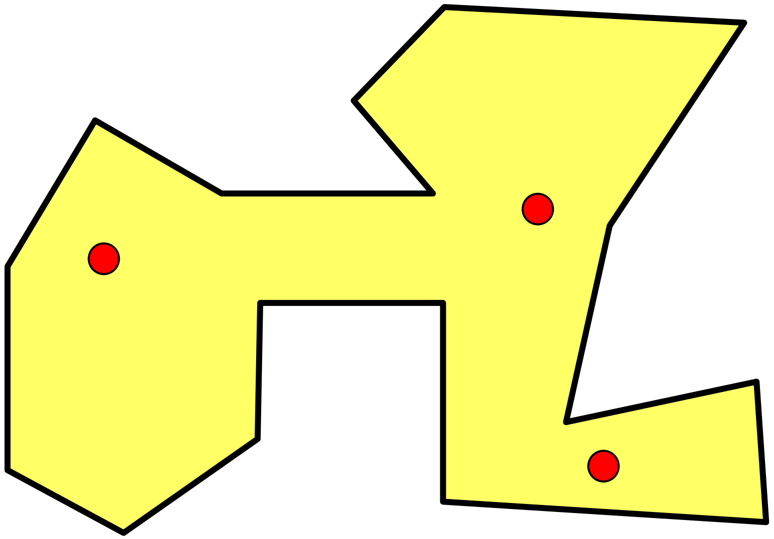


Guarding

- **Visibility Problems: Guarding** and **Hiding**

- Input: simple polygon P

  - **Guarding:** find a minimum number of points (guards) in P, such that each point in P is seen by at least one guard



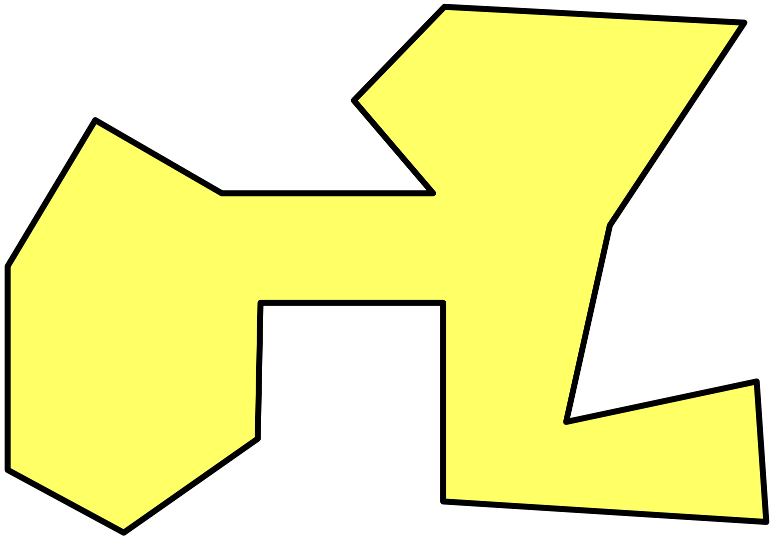
Hiding

- **Hiding:** find a maximum number of points in P, such that no two of these points see each other

Shermer, 89

Hiding

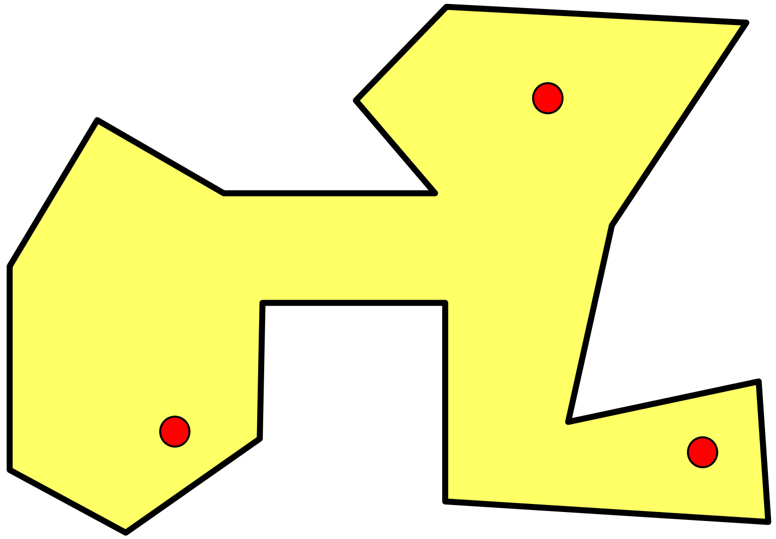
- **Hiding:** find a maximum number of points in P, such that no two of these points see each other



Shermer, 89

Hiding

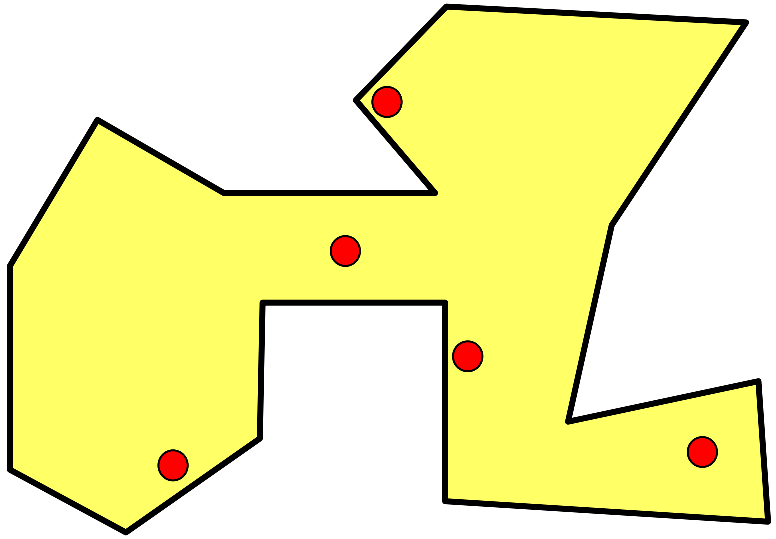
- **Hiding:** find a maximum number of points in P, such that no two of these points see each other



Shermer, 89

Hiding

- **Hiding:** find a maximum number of points in P, such that no two of these points see each other



Shermer, 89

Hiding

- **Maximum Hidden Set** (**MHS**) problem :

  asks for a set S of maximum cardinality of points in a given polygon, such that no two points in S see each other

- **Maximum Hidden Vertex Set** (**MHVS**) problem:

  asks for a set S of maximum cardinality of vertices of a given polygon, such that no two vertices in S see each other

- **MHS** and **MHVS** problems are **NP-hard** for arbitrary and for orthogonal polygons

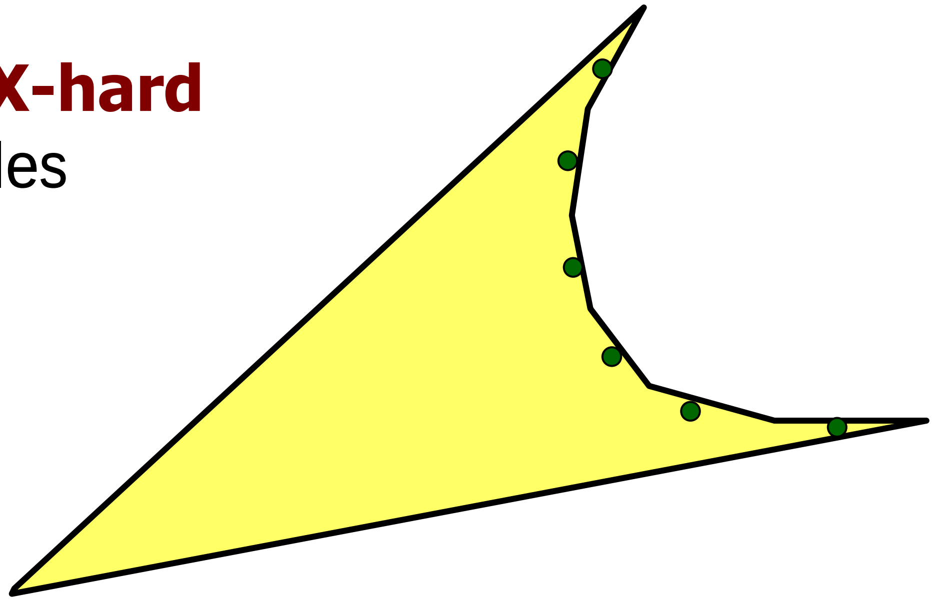
Shermer, 89

Hiding

- **Maximum Hidden Set (MHS)**
- **Maximum Hidden Vertex Set (MHVS)**

(In-) Approximability Eidenbenz, 2000

MHS y **MHVS** are **APX-hard**
for polygons without holes



The best approximating
algorithm achieves ratio $\Theta(n)$

Hiding (algorithmic problem)

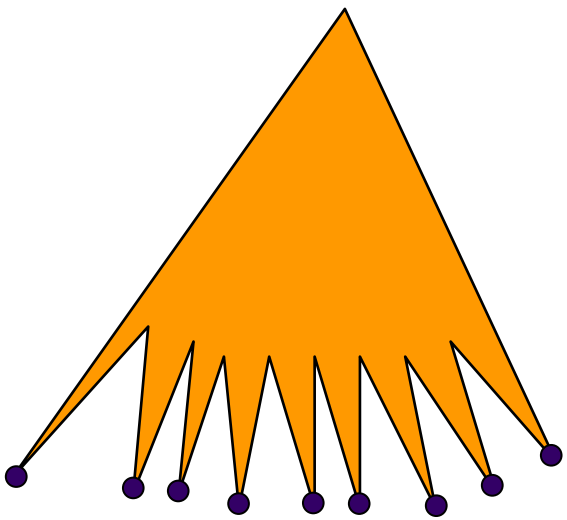
Let P be a polygon and H a set of vertices in P. We say that H is a **hidden vertex set** if no two vertices in H see each other

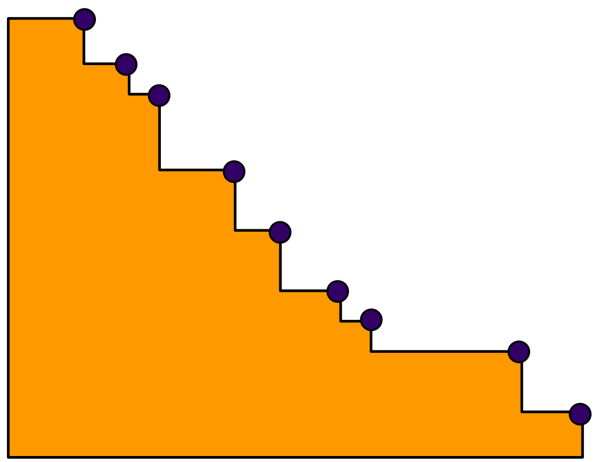
Given a polygon P, with n vertices,

determine H of maximum cardinality

Hiding (combinatorial problem)

- The size of the maximum vertex hidden set of a polygon with **n** vertices is at most $\lceil n/2 \rceil$
- The size of the maximum vertex hidden set of an orthogonal polygon with **n** vertices is at most $(n-2)/2$


Saw polygons


Staircase polygons

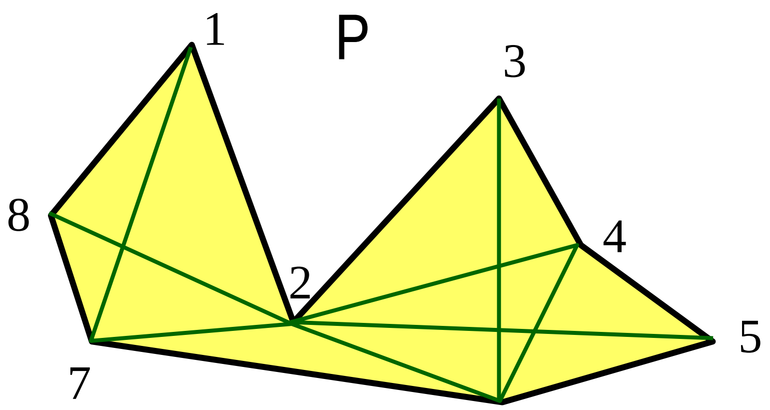
Approximation Algorithms

- Propose **approximation algorithms** to compute solutions for the **MHVS** problem on polygons (arbitrary and orthogonal)
  - Greedy constructive algorithms: A_1 and A_2
  - Two based on the general metaheuristics Simulated Annealing and Genetic Algorithms: M_1 and M_2
- Realize a comparative study of the solutions obtained by the different algorithms
- Determine the **approximation ratio** of our algorithms

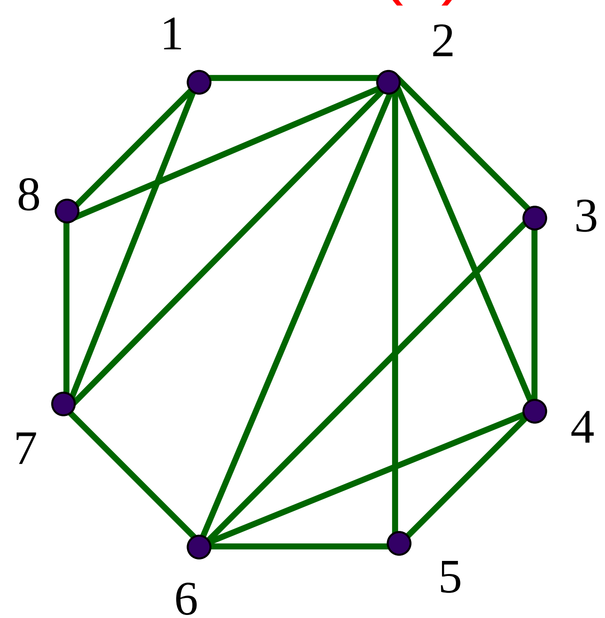
Approximation Algorithms: Preprocessing

Visibility graph of P, **VG(P)**

The nodes of **VG(P)** are the vertices of P, and there is an edge between the vertices a and b if a sees b



VG(P)



Greedy Constructive Algorithms

Natural approach to find H is to do it greedily:

- start with an empty set
- add hidden vertices one by one until H is achieved

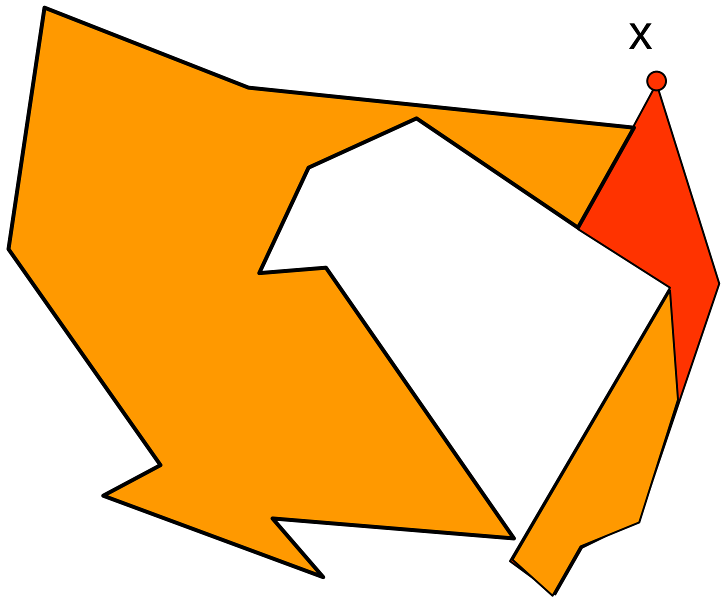
  selecting at each step a hidden vertex from the set of vertices of P according to some rule

We used two rules:

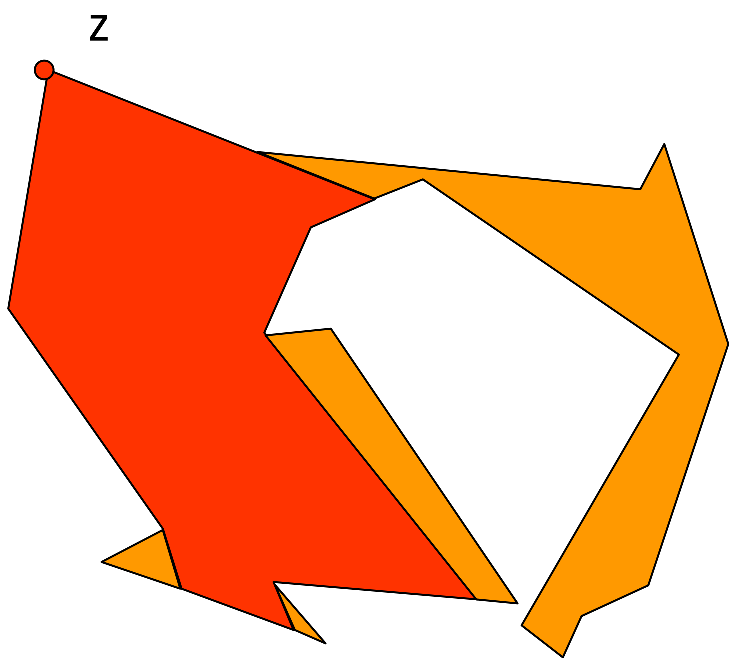
- The first rule is based in the **hidden region** concept
- The second rule is based in the number of vertices seen by each vertex

Greedy Algorithms: A_1

VisP(x) is the visibility polygon of x



2 hidden regions for x



4 hidden regions for z

Greedy Algorithms

We select vertices one to one, according to

- **A_1**

  highest number of hidden regions

- **A_2**

  lesser number visible vertices

Algorithms based in Metaheuristics

A metaheuristic is a set of concepts that can be used to define heuristic methods which can be applied to a wide set of different optimization problems.

- Simulated Annealing (SA)
- Iterated Local Search (ILS)
- Tabu Search (TS).
- Genetic Algorithms (GA)
- Ant Colony Optimization (ACO)
- ...

Simulated Annealing: Overview

SA tries to minimize the limitation of the local (maximizaton) search algorithms, which stop as soon as they find a local maximum

- allows to accept solutions of worse quality than the current solution (downhill moves) with a certain probability

Fundamental idea

- If the new solution (neighbour solution) is better (high cost) than the actual solution, this new solution is accepted
- If the new solution is worse (low cost) than the actual solution, this new solution can be accepted with a given probability
- This probability is dependent of a parameter called Temperature (T), which decreases over the algorithm iterations according to a decrement rule.

Simulated Annealing: Overview

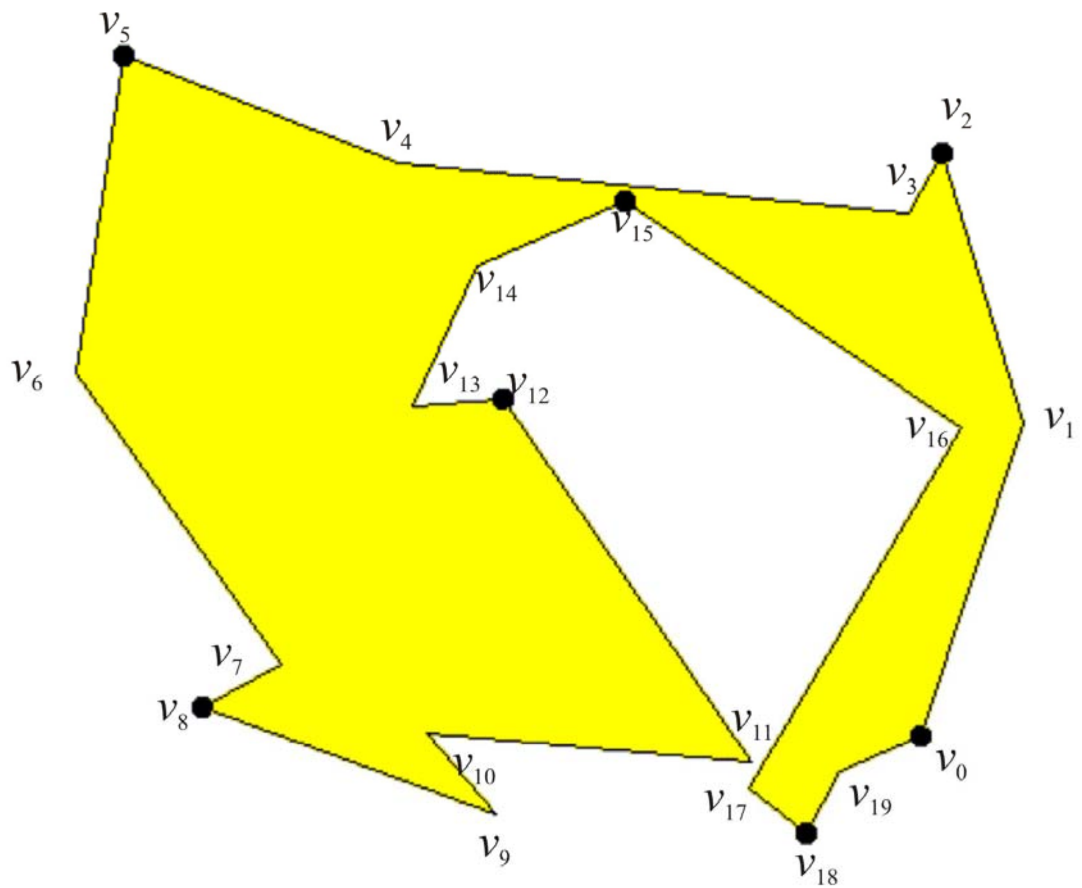
Specific Parameters
(of the problem)

- Solution Space (set S)
- Cost or Objective Function, C
- Neighbourhood of each solution
- Initial Solution

Generic Parameters
(of the annealing strategy)

- Initial temperature (T_0)
- Temperature Decrement Rule
- Number of iterations in each temperature, $N(T)$
- Termination condition

M_1: Specific Parameters (Solution Space)



	v^i_0	v^i_1	v^i_2																	v^i_{19}
S_i	1	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0

M_1: Specific Parameters (Cost)

- **Cost or Objective Function**

  $f : S \rightarrow N$ $\qquad$ $f(S_i)$ = number of 1's in S_i

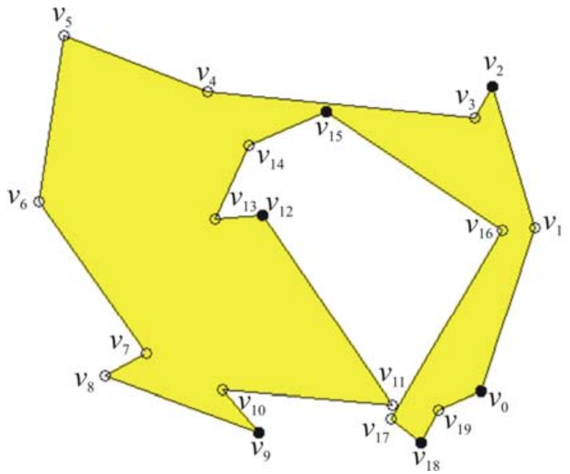
M_1: Specific Parameters (Neighbourhood)

Given $S_i = v_0^i v_1^i \ldots v_{n-1}^i$ we randomly generate a natural number $t \in [0, n-1]$ and then

- If $v_t^i = 1$ then we make $v_t^{i+1} = 0$ and accept this new solution, S_{i+1}, with probability

- If $v_t^i = 0$ then we make $v_t^{i+1} = 1$ and
  - If S_{i+1} is a valid solution we accept it
  - If S_{i+1} is not a valid solution we **validate** it (i.e., we mark all hidden vertices as not hidden if v_t sees them) and accept this new solution with probability

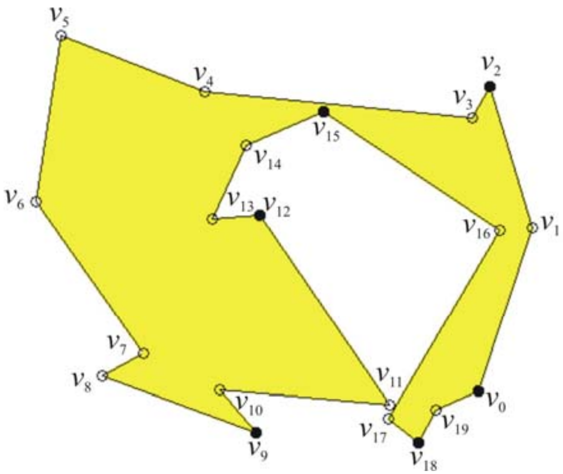
M_1 : Specific Parameters (Neighbourhood)

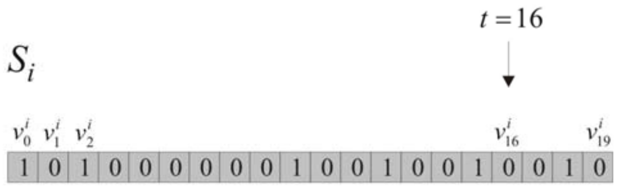


S_i

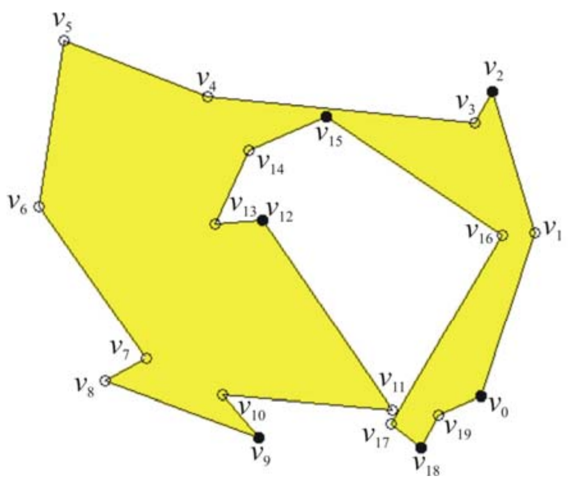
v_0^i	v_1^i	v_2^i														v_{16}^i			v_{19}^i
1	0	1	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0

M_1 : Specific Parameters (Neighbourhood)





M_1 : Specific Parameters (Neighbourhood)



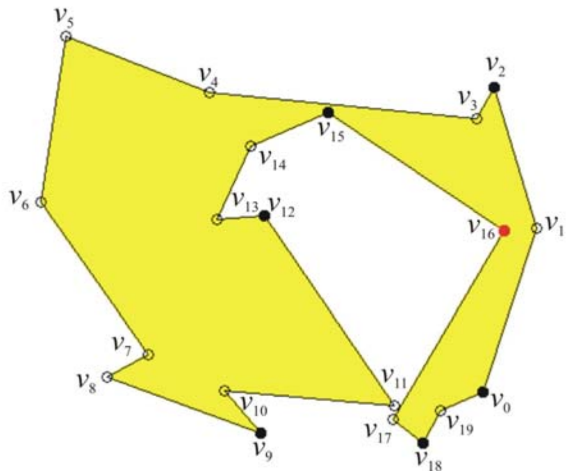
S_i

$t = 16$

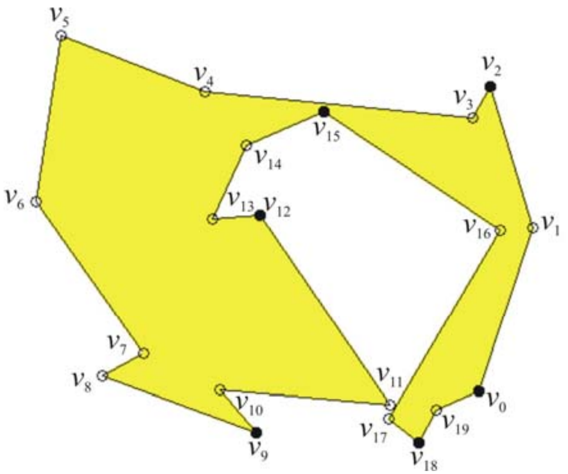
v_0^i	v_1^i	v_2^i														v_{16}^i			v_{19}^i
1	0	1	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0

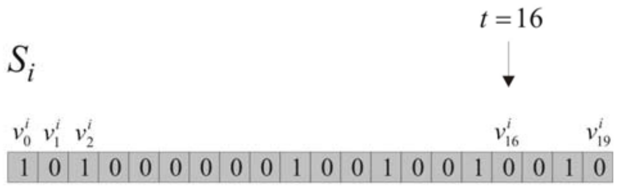
S_{i+1}

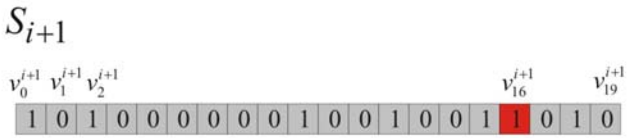
v_0^{i+1}	v_1^{i+1}	v_2^{i+1}														v_{16}^{i+1}			v_{19}^{i+1}
1	0	1	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	1	0

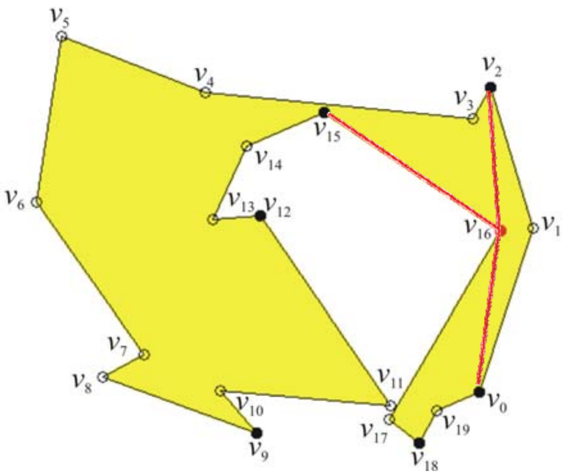


M_1 : Specific Parameters (Neighbourhood)

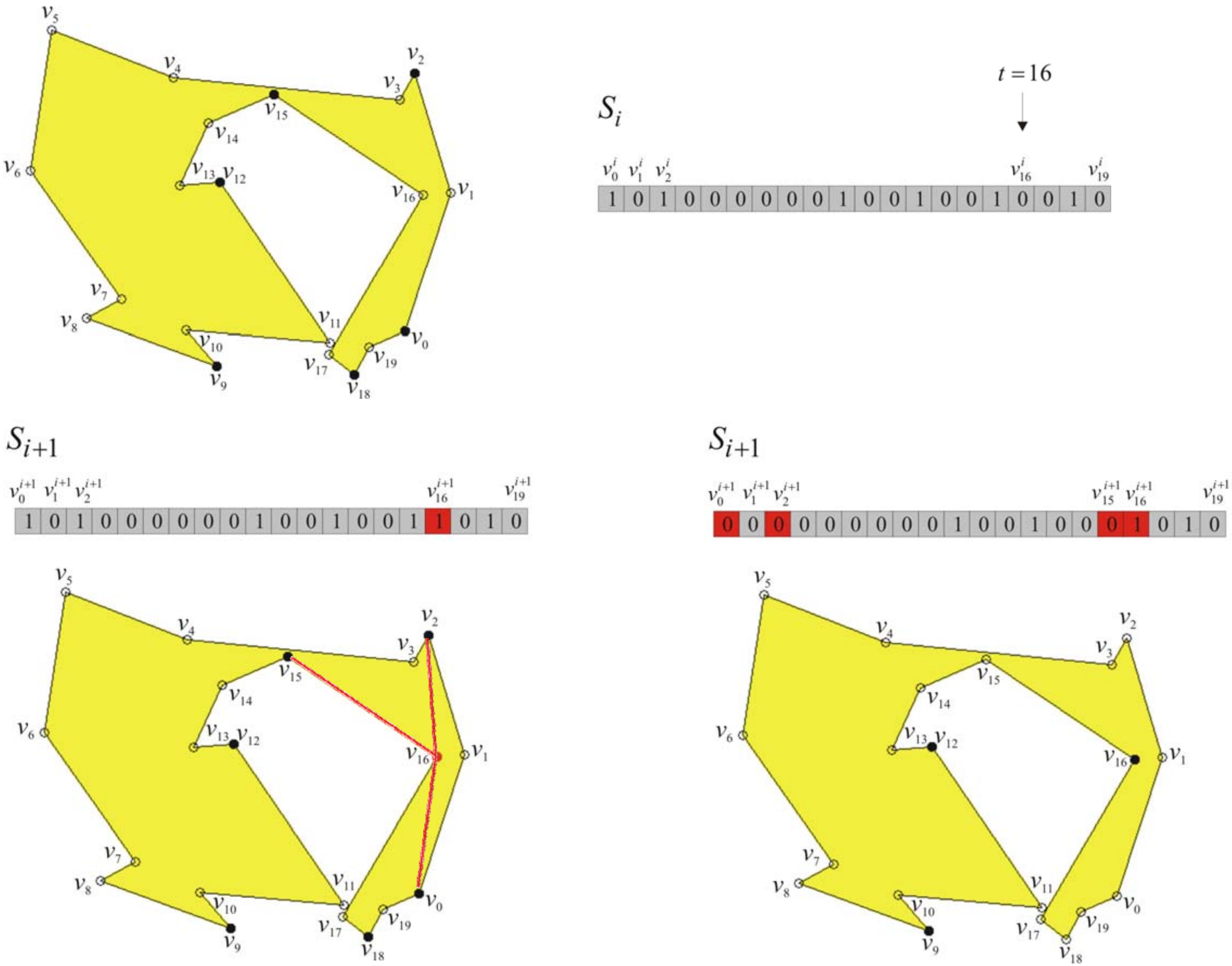








M$_1$: Specific Parameters (Neighbourhood)

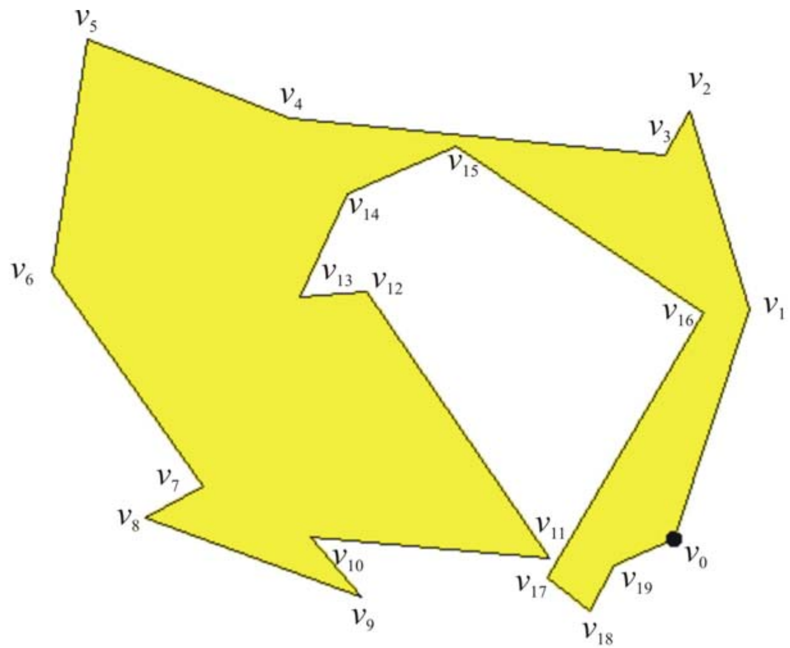


M_1: Generic Parameters (Initial Solution)

Initial Solution

$S_0 = 10\ldots00$

v_0 is marked as hidden the remainder are labeled not hidden



	v_0^0	v_1^0																		v_{19}^0
S_0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_1: Generic Parameters (T_0 & Decrement Rule)

- **Initial Temperature, T_0**

  We realize a comparative study taking into account two different types of T_0 :

  (1) $T_0 = n$ (dependent on the number of vertices of the polygon)

  (2) $T_0 = 1000.0$ (constant)

- **Temperature Decrement Rule**

  Three different types of decrement rules:

  (1) $T_{k+1} = \dfrac{T_0}{1+k}$ (FSA decrease)

  (2) $T_{k+1} = \dfrac{T_0}{e^k}$ (VFSA decrease)

  (3) $T_{k+1} = \alpha T_k$, where $0 < \alpha < 1$ (geometric decrease)

M_1: Generic Parameters ($N(T)$ & Termination condition)

- **Number of iterations in each temperature**

$$N(T) = T$$

more iterations for high temperatures, which will be when the solutions are far to the optimum

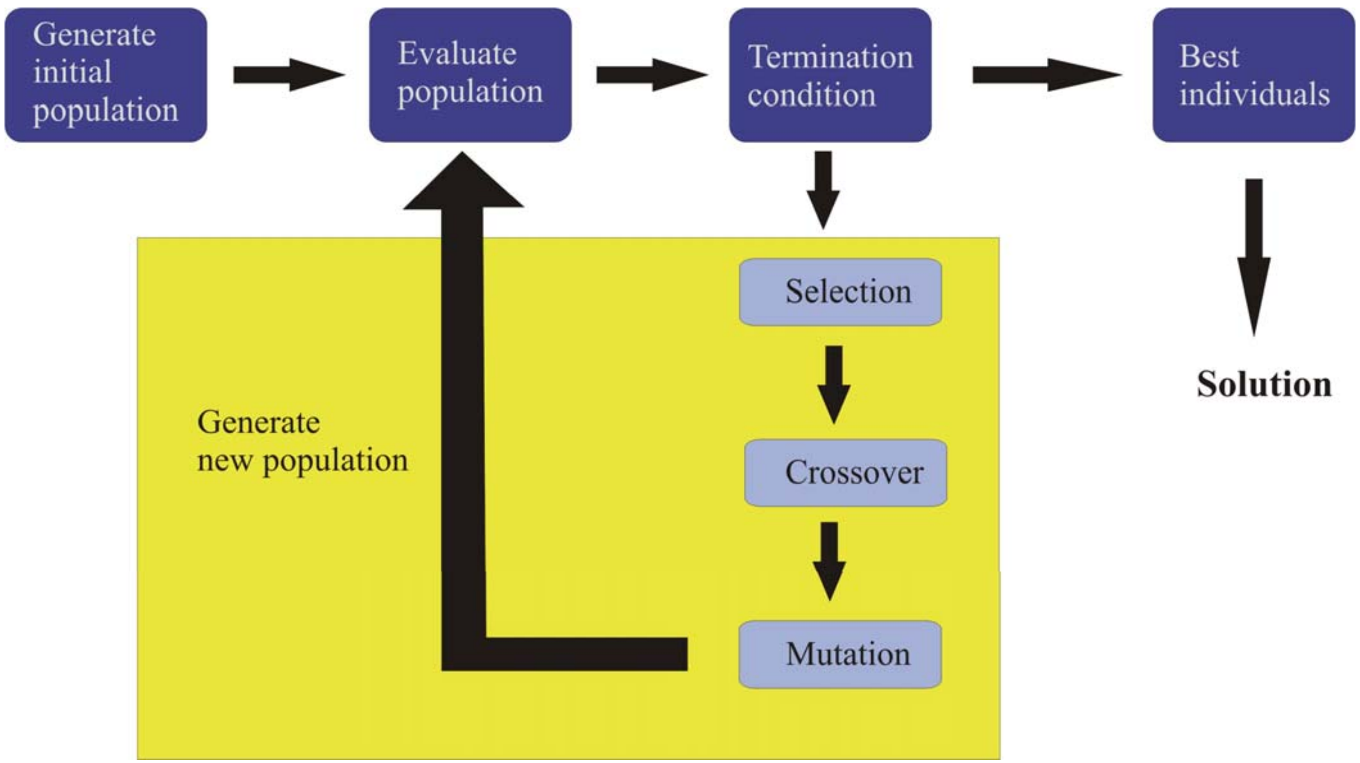
- **Termination Condition**

We choose to stop when $T \leq 0.005$

Theoretically, the search should stop when $T = 0$. But, normally, it is possible to finish with a temperature greater then zero, without quality loss in the solution

Genetic Algorithms: Overview

- Are methods that simulate, through algorithms, the processes of the natural evolution (biological)



Genetic Algorithms: Overview

- A genetic representation of the possible solutions, individuals or chromosomes, to the problem (**Encoding**)
- **Initial Population**
- A function to evaluate each individual (**Objective or Fitness function**)
- Genetic operators (**Selection**, **Crossover**, and **Mutation**)
- Other parameters Population's Size, Probability of the operators, Population's Evaluation, Population's Generation, Termination Condition)

M_2: Parameters (Encoding)

Encoding

An individual **I** is a hidden vertex set for P

$$\mathbf{I} = g_0 g_1 \ldots g_{n-1},$$

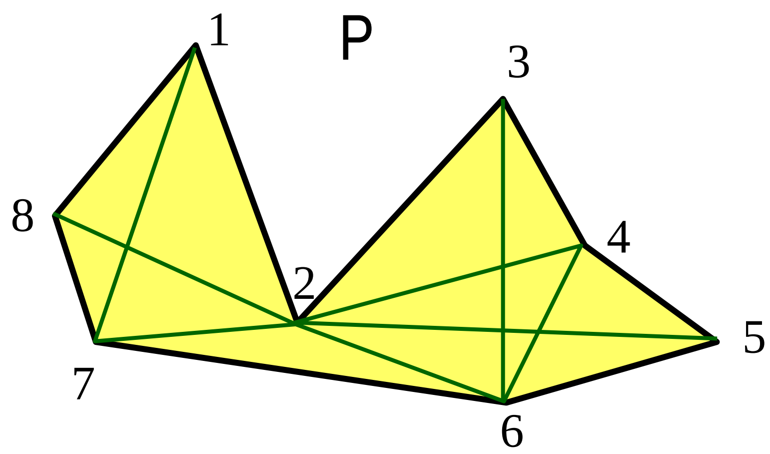
g_i is a gene and represents the vertex v_i

- If $g_i = 0$ the vertex v_i is marked as not hidden
- If $g_i = 1$ the vertex v_i is marked as hidden

M2: Parameters (Initial Population)

Initial Population

- The size of the population is n



v_1	10101000	v_5	00101001
v_2	01000000	v_6	00000101
v_3	00101010	v_7	00100010
v_4	00010010	v_8	00101001

M_2: Parameters (Fitness function & Selection)

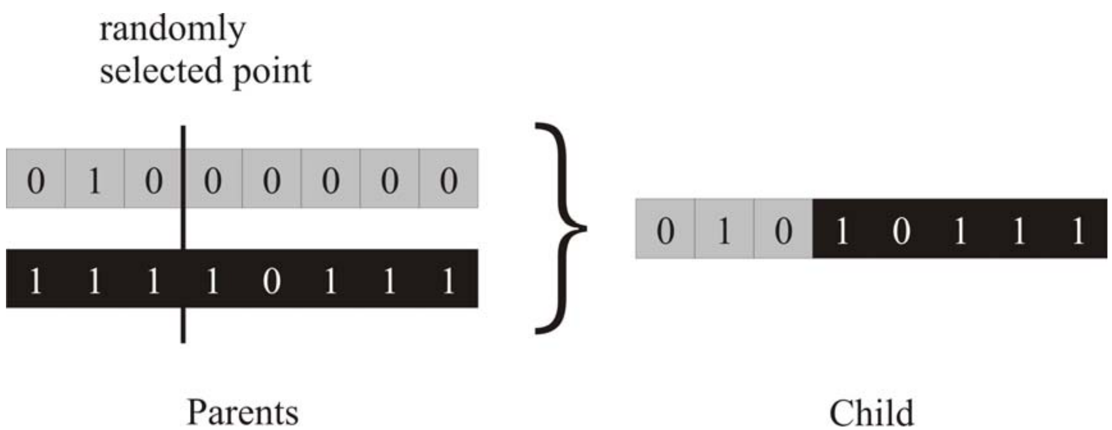
- **Objective or Fitness Function**

  f(I) = number of 1's in I

- **Selection**
  - The best individuals should be chosen to be reproduced
  - We use the roulette wheel selection

Crossover

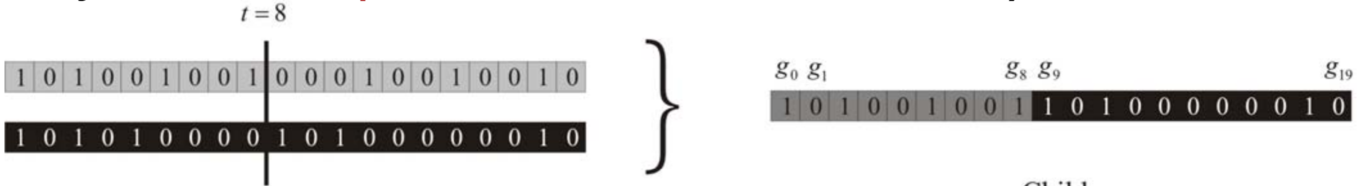
- operates in selected genes of the parents and create new individuals (children)
- Single point crossover, to generate one child



- Crossover occurs with a given probability $pc = 0.9$

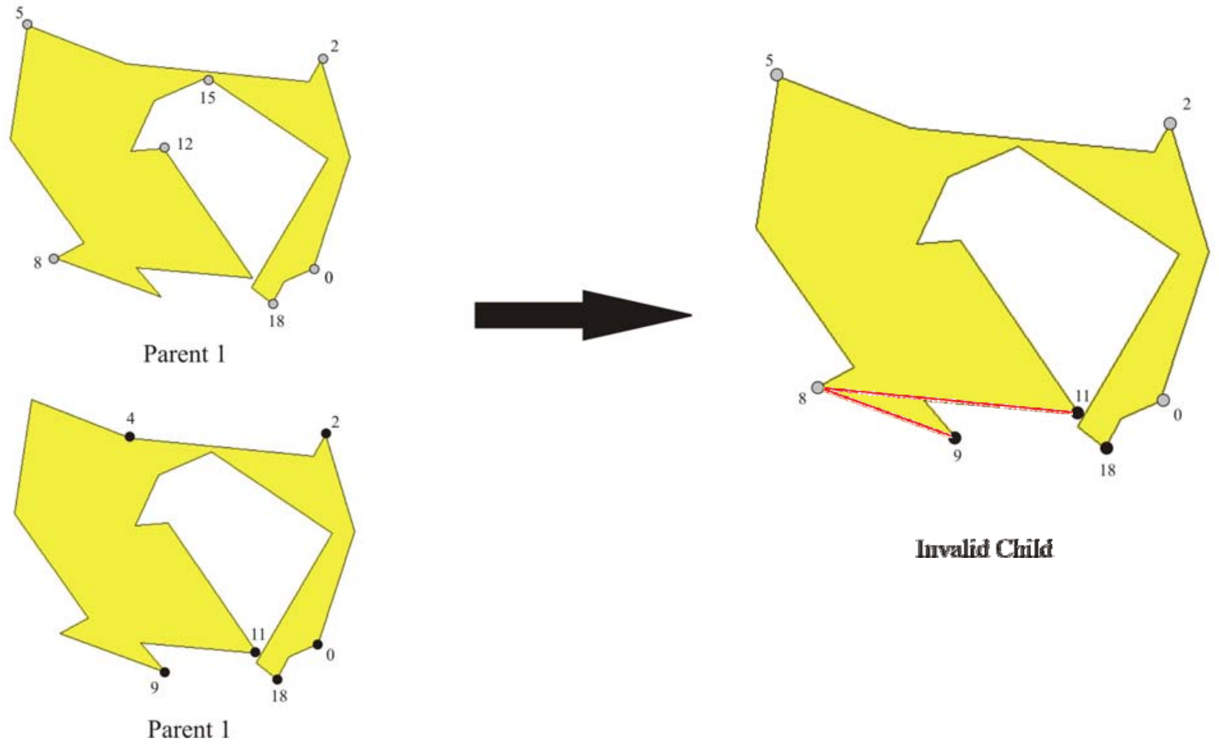
M_2: Parameters (Crossover)

The child resulting from this crossover may not be valid (i.e., it may not correspond to a hidden vertex set)

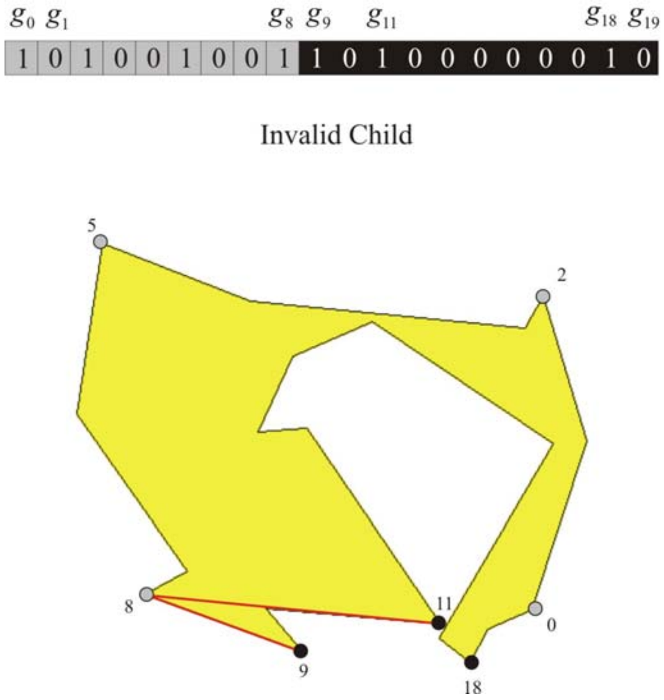


Parents

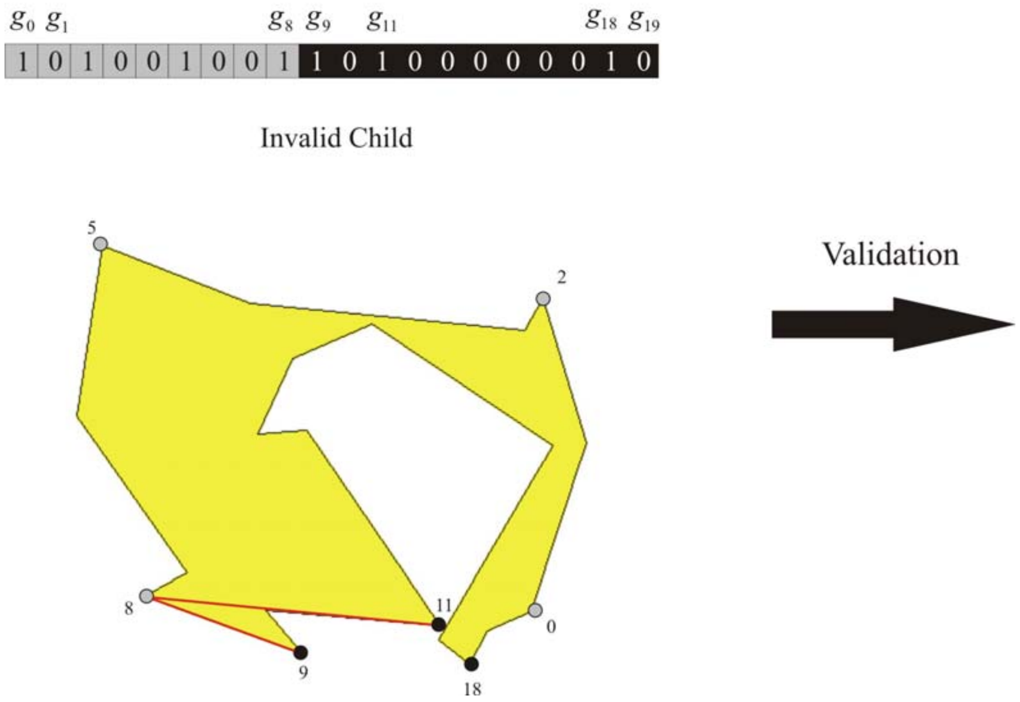
Child



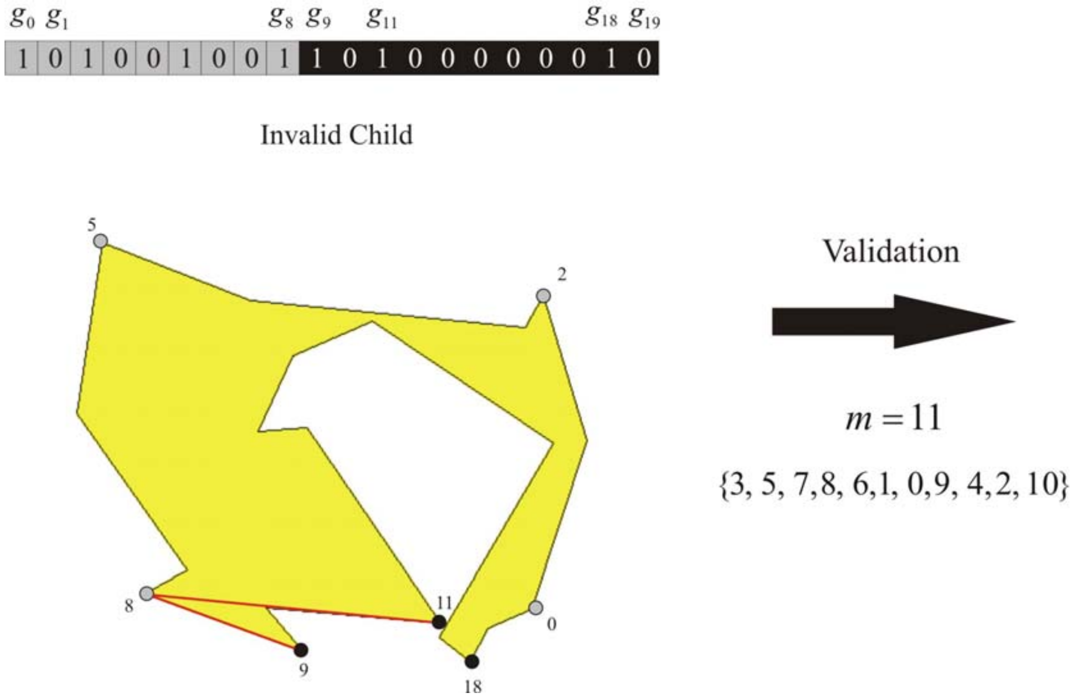
M_2: Parameters (Crossover validation)



M_2: Parameters (Crossover validation)



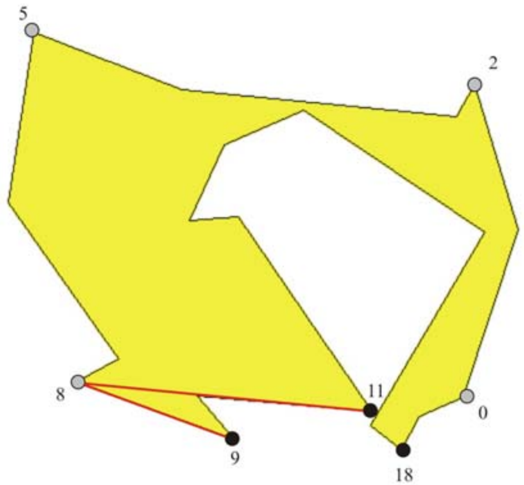
M_2: Parameters (Crossover validation)



M_2: Parameters (Crossover validation)

g_0	g_1							g_8	g_9		g_{11}							g_{18}	g_{19}
1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	1	0

Invalid Child



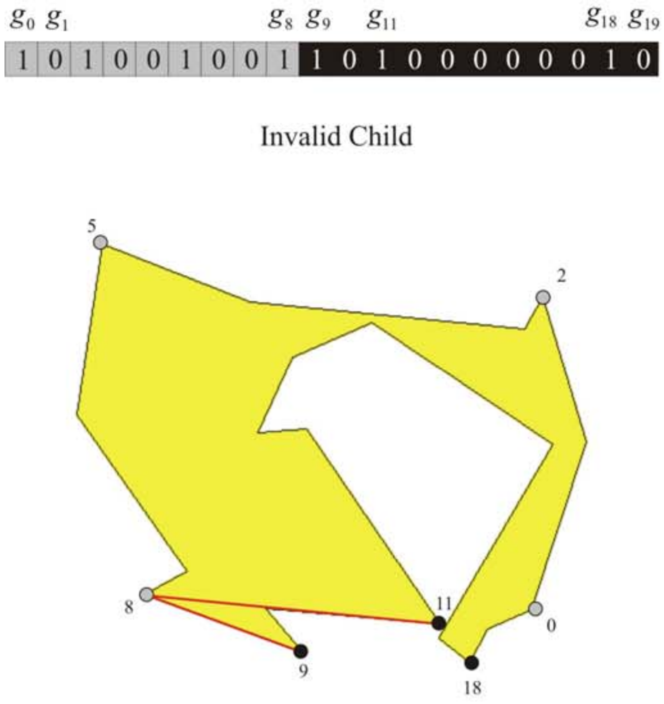
Validation

$$m = 11$$

$$\{3, 5, 7, 8, 6, 1, 0, 9, 4, 2, 10\}$$

$$g_{8+3+1} = g_{12} = 0$$
$$g_{8+5+1} = g_{14} = 0$$
$$g_{8+7+1} = g_{16} = 0$$
$$g_{8+8+1} = g_{17} = 0$$
$$g_{8+6+1} = g_{15} = 0$$
$$g_{8+1+1} = g_{10} = 0$$
$$g_{8+0+1} = g_9 = 1$$
$$g_{8+9+1} = g_{18} = 1$$
$$g_{8+4+1} = g_{13} = 0$$
$$g_{8+2+1} = g_{11} = 1$$
$$g_{8+10+1} = g_{19} = 0$$

M_2: Parameters (Crossover validation)

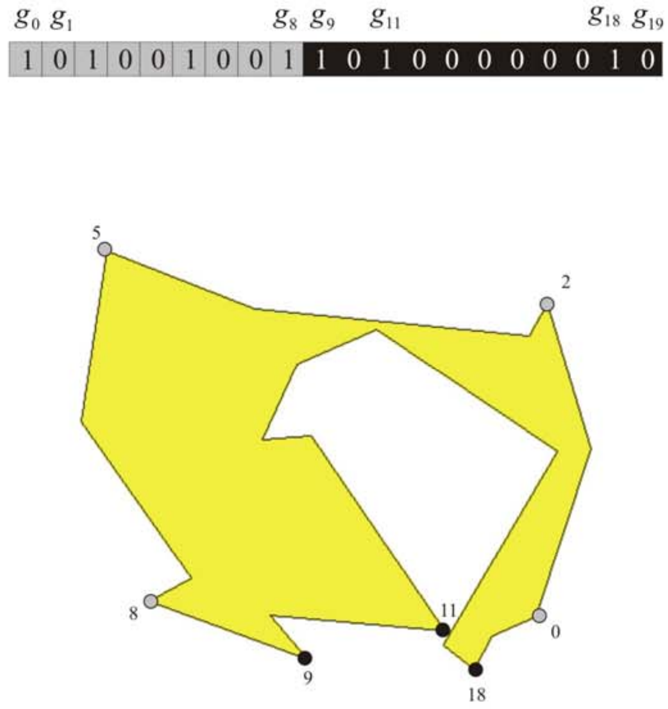


Validation

$m = 11$

$\{3, 5, 7, 8, 6, 1, 0, 9, 4, 2, 10\}$

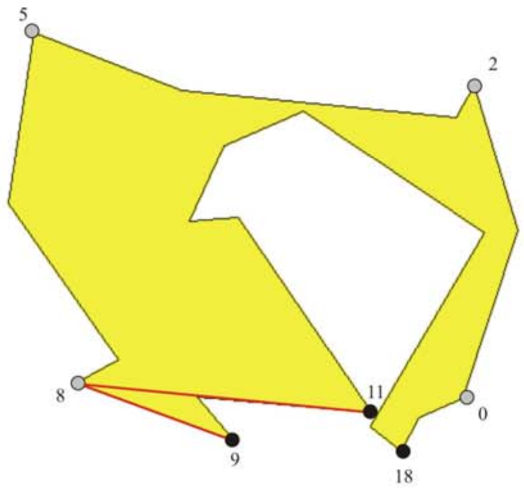
$$g_{8+3+1} = g_{12} = 0$$
$$g_{8+5+1} = g_{14} = 0$$
$$g_{8+7+1} = g_{16} = 0$$
$$g_{8+8+1} = g_{17} = 0$$
$$g_{8+6+1} = g_{15} = 0$$
$$g_{8+1+1} = g_{10} = 0$$
$$g_{8+0+1} = g_9 = 1$$
$$g_{8+9+1} = g_{18} = 1$$
$$g_{8+4+1} = g_{13} = 0$$
$$g_{8+2+1} = g_{11} = 1$$
$$g_{8+10+1} = g_{19} = 0$$



M_2 : Parameters (Crossover validation)

g_0	g_1							g_8	g_9		g_{11}							g_{18}	g_{19}
1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	1	0

Invalid Child



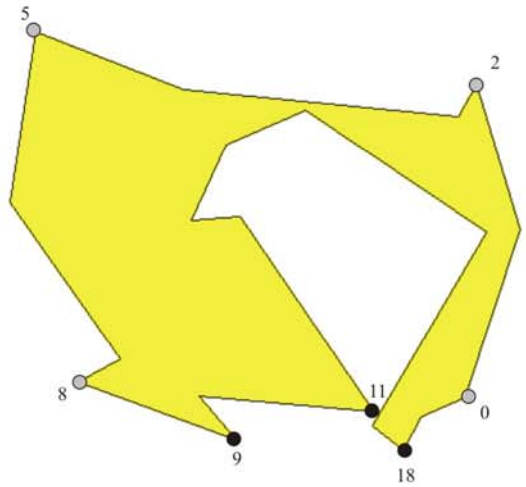
Validation

$$m = 11$$

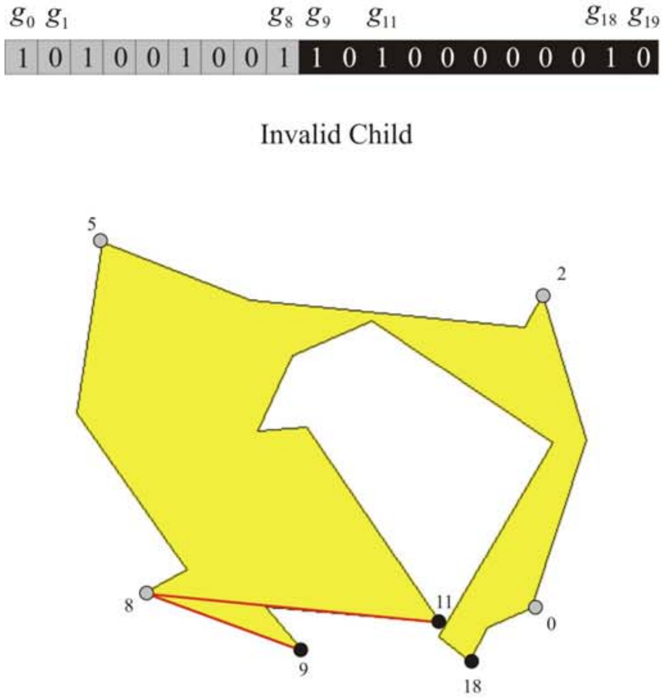
$$\{3, 5, 7, 8, 6, 1, 0, 9, 4, 2, 10\}$$

$$g_{8+3+1} = g_{12} = 0$$
$$g_{8+5+1} = g_{14} = 0$$
$$g_{8+7+1} = g_{16} = 0$$
$$g_{8+8+1} = g_{17} = 0$$
$$g_{8+6+1} = g_{15} = 0$$
$$g_{8+1+1} = g_{10} = 0$$
$$\boxed{g_{8+0+1} = g_9 = 1}$$
$$g_{8+9+1} = g_{18} = 1$$
$$g_{8+4+1} = g_{13} = 0$$
$$g_{8+2+1} = g_{11} = 1$$
$$g_{8+10+1} = g_{19} = 0$$

g_0	g_1							g_8	g_9		g_{11}							g_{18}	g_{19}
1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	1	0



M_2 : Parameters (Crossover validation)



Validation

$m = 11$

$\{3, 5, 7, 8, 6, 1, 0, 9, 4, 2, 10\}$

$g_{8+3+1} = g_{12} = 0$

$g_{8+5+1} = g_{14} = 0$

$g_{8+7+1} = g_{16} = 0$

$g_{8+8+1} = g_{17} = 0$

$g_{8+6+1} = g_{15} = 0$

$g_{8+1+1} = g_{10} = 0$

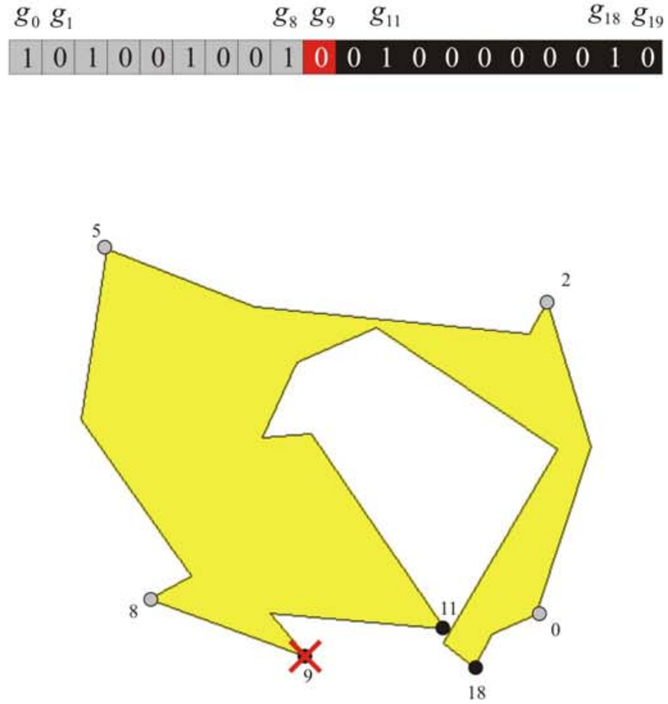
$g_{8+0+1} = g_{9} = 1$

$g_{8+9+1} = g_{18} = 1$

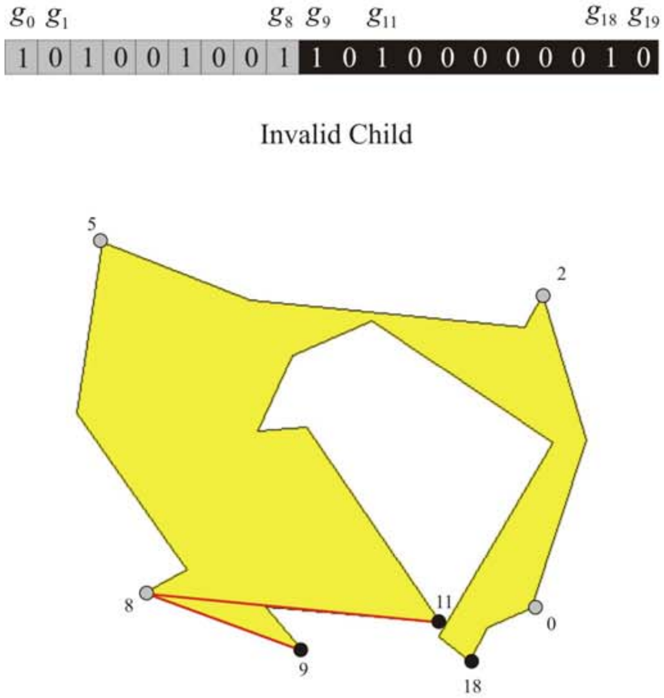
$g_{8+4+1} = g_{13} = 0$

$g_{8+2+1} = g_{11} = 1$

$g_{8+10+1} = g_{19} = 0$



M_2 : Parameters (Crossover validation)

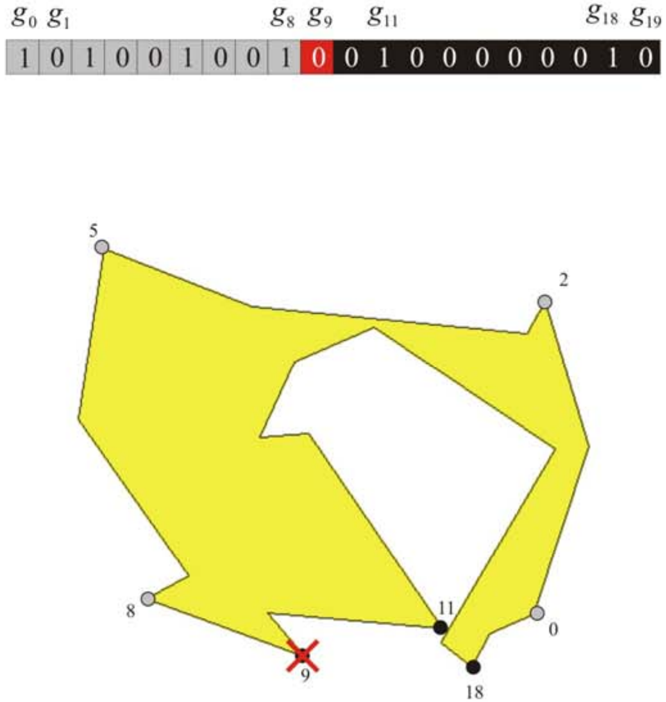


Validation

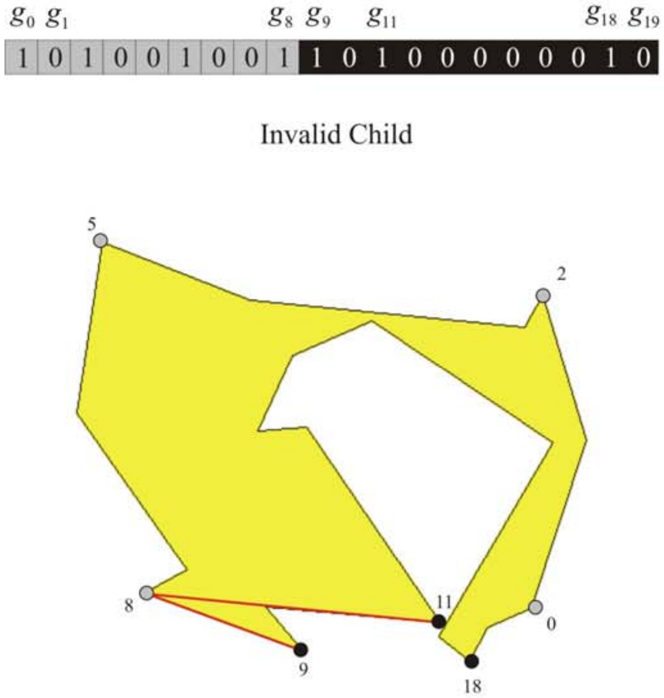
$$m = 11$$

$$\{3, 5, 7, 8, 6, 1, 0, 9, 4, 2, 10\}$$

$$g_{8+3+1} = g_{12} = 0$$
$$g_{8+5+1} = g_{14} = 0$$
$$g_{8+7+1} = g_{16} = 0$$
$$g_{8+8+1} = g_{17} = 0$$
$$g_{8+6+1} = g_{15} = 0$$
$$g_{8+1+1} = g_{10} = 0$$
$$g_{8+0+1} = g_{9} = 1$$
$$\boxed{g_{8+9+1} = g_{18} = 1}$$
$$g_{8+4+1} = g_{13} = 0$$
$$g_{8+2+1} = g_{11} = 1$$
$$g_{8+10+1} = g_{19} = 0$$



M_2 : Parameters (Crossover validation)

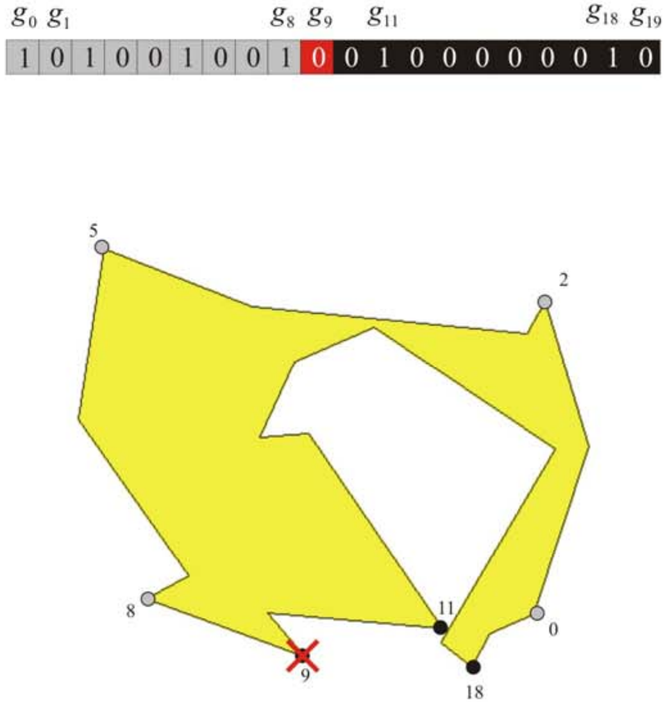


Validation

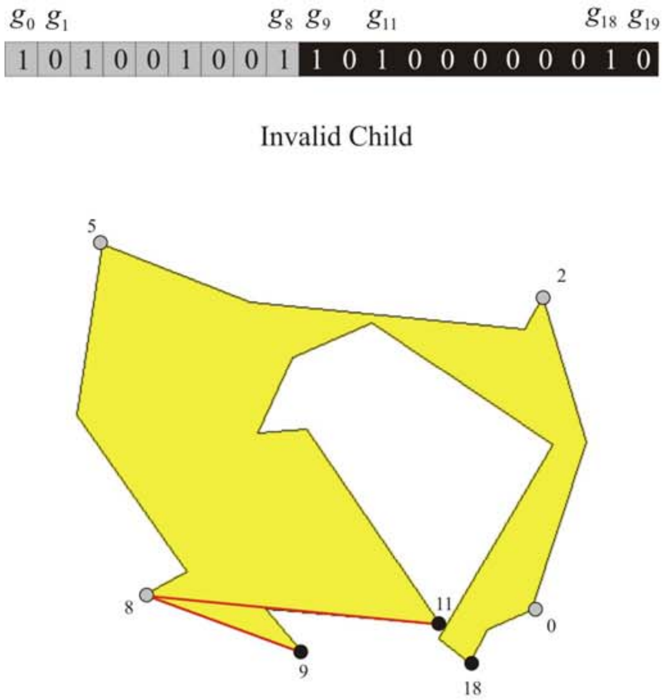
$$m = 11$$

$$\{3, 5, 7, 8, 6, 1, 0, 9, 4, 2, 10\}$$

$$g_{8+3+1} = g_{12} = 0$$
$$g_{8+5+1} = g_{14} = 0$$
$$g_{8+7+1} = g_{16} = 0$$
$$g_{8+8+1} = g_{17} = 0$$
$$g_{8+6+1} = g_{15} = 0$$
$$g_{8+1+1} = g_{10} = 0$$
$$g_{8+0+1} = g_{9} = 1$$
$$g_{8+9+1} = g_{18} = 1$$
$$g_{8+4+1} = g_{13} = 0$$
$$\boxed{g_{8+2+1} = g_{11} = 1}$$
$$g_{8+10+1} = g_{19} = 0$$



M_2 : Parameters (Crossover validation)

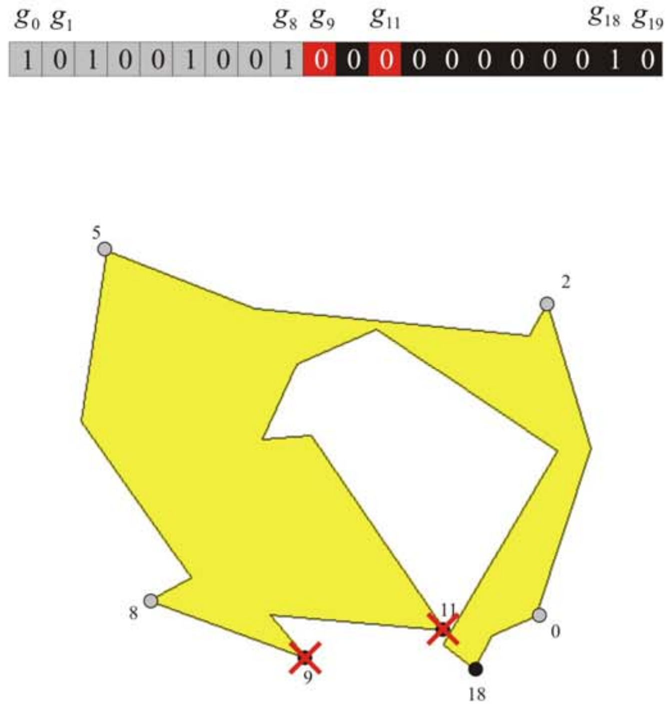


Validation

$$m = 11$$

$$\{3, 5, 7,8, 6,1, 0,9, 4,2, 10\}$$

$$g_{8+3+1} = g_{12} = 0$$
$$g_{8+5+1} = g_{14} = 0$$
$$g_{8+7+1} = g_{16} = 0$$
$$g_{8+8+1} = g_{17} = 0$$
$$g_{8+6+1} = g_{15} = 0$$
$$g_{8+1+1} = g_{10} = 0$$
$$g_{8+0+1} = g_{9} = 1$$
$$g_{8+9+1} = g_{18} = 1$$
$$g_{8+4+1} = g_{13} = 0$$
$$\boxed{g_{8+2+1} = g_{11} = 1}$$
$$g_{8+10+1} = g_{19} = 0$$



M_2 : **Parameters** (Crossover validation)

g_0 g_1 ... g_8 g_9 ... g_{11} ... g_{18} g_{19}

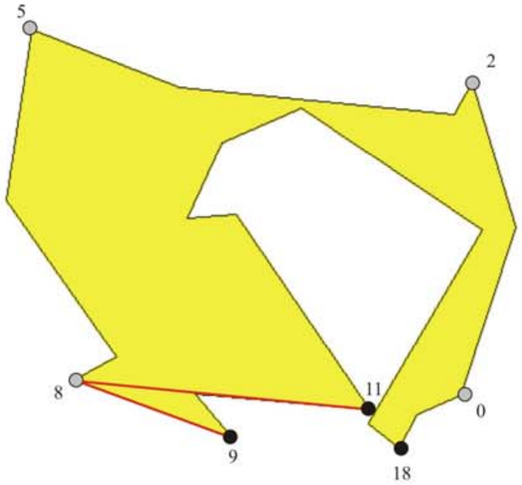
1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	1	0

Invalid Child

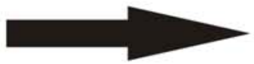
g_0 g_1 ... g_8 g_9 ... g_{11} ... g_{18} g_{19}

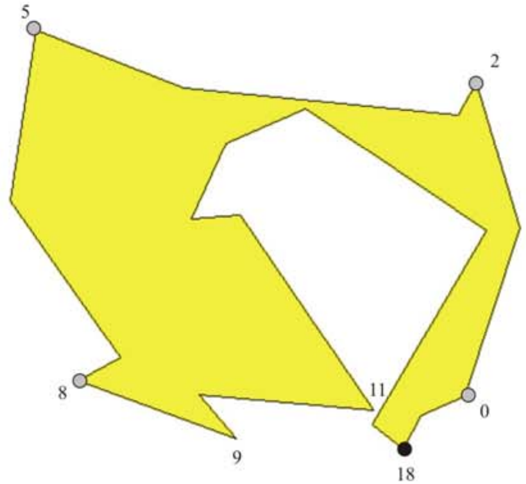
1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0

Valid Child



Validation





$$m = 11$$

$$\{3, 5, 7, 8, 6, 1, 0, 9, 4, 2, 10\}$$

$$g_{8+3+1} = g_{12} = 0$$
$$g_{8+5+1} = g_{14} = 0$$
$$g_{8+7+1} = g_{16} = 0$$
$$g_{8+8+1} = g_{17} = 0$$
$$g_{8+6+1} = g_{15} = 0$$
$$g_{8+1+1} = g_{10} = 0$$
$$g_{8+0+1} = g_{9} = 1$$
$$g_{8+9+1} = g_{18} = 1$$
$$g_{8+4+1} = g_{13} = 0$$
$$g_{8+2+1} = g_{11} = 1$$
$$g_{8+10+1} = g_{19} = 0$$

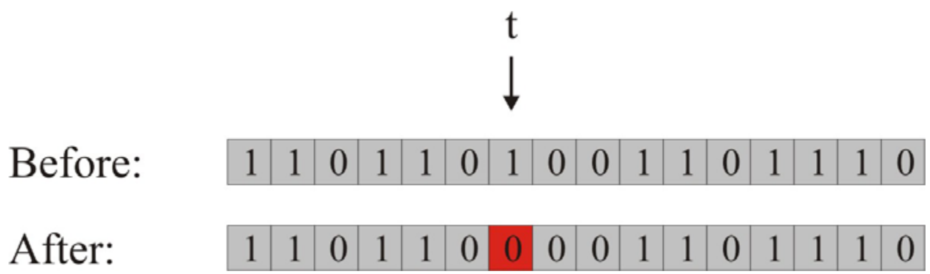
M_2 : Parameters (Mutation)

Mutation

probability $pm = 0.05$

randomly generate a natural number $0 \leq t \leq n-1$

- If $g_t=1$ then we change its value to 0
- If $g_t=0$ we change its value to 1 only if the resultant individual is valid



M_2 : Parameters (Population's Generation and Fitness & T. Cond.)

- **Population's Generation**

We replaced the worst individual of the population by the child obtained at the crossover.

- **Population's Evaluation or Population's Fitness,**

F(P(t)) = max {fitness of individuals}

- **Termination Condition**

If the fitness of the populations remains the same for a large number of iterations, h, we can assume that we are close to the optimal

$h = 5000$

Experiments & Results

- Computational Geometry Algorithms Library (CGAL)
- We realized our experiments on a large set of randomly generated polygons
  - General polygons

    generated using the CGAL's function random_polygon_2
  - Orthogonal polygons

    we used the polygon generator developed by O'Rourke
- Four sets of polygons, each one with 50 polygons of 50, 100, 150 and 200 vertices

Experiments & Results: SA's Parameters (Arbitrary polygons)

- Initial Temperature, T_0

(1) $T_0 = n$

(2) $T_0 = 1000.0$

- Temperature Decrement Rule

(1) $T_{k+1} = \frac{T_0}{1+k}$ (FSA decrease)

(2) $T_{k+1} = \frac{T_0}{e^k}$ (VFSA decrease)

(3) $T_{k+1} = \alpha T_k$, where $0 < \alpha < 1$ (geometric decrease)

The choice of $T_0 = n$ and FSA is the best one.

Experiments & Results: Four Algorithms (Arbitrary polygons)

Results obtained with M_1

Vertices	PP (seconds)	$\|H\|$	Time (seconds)	Iterations
50	0.48	13.96	0.04	9999
100	2.42	27.4	0.1	19999
150	7.2	40.5	0.26	29999
200	15.58	53.86	0.36	39999

Results obtained with A_1

Vertices	PP (seconds)	$\|H\|$	Time (seconds)	Iterations
50	0.48	12.1	0.08	12.1
100	2.42	24.18	0.46	24.18
150	7.2	35.12	0.5	35.12
200	15.58	46.62	0.9	46.62

Results obtained with A_2

Vertices	PP (seconds)	$\|H\|$	Time (seconds)	Iterations
50	0.48	13.58	0	13.58
100	2.42	27.12	0	27.12
150	7.2	39.88	0	39.88
200	15.58	52.68	0	52.68

Results obtained with M_2

Vertices	PP (seconds)	$\|H\|$	Time (seconds)	Iterations
50	0.48	13.54	0.06	5226.4
100	2.42	26.28	0.08	5680.0
150	7.2	38.3	0.26	6666.7
200	15.58	50.34	0.5	7160.2

General conclusions

The algorithm M_1 seems to be the best one, since:

- the obtained average of hidden vertices is better than the others; and
- in spite of the average of the number of iterations is the biggest, the only algorithm that is faster than it, is the A_2

Experiments & Results: M_1 Heuristic (Arbitrary polygons)

Vertices	20	40	60	80	100	120	140	150	200
$\|H\|$	5.7	10.98	16.3	21.58	26.88	32.08	37.2	39.7	53.22

Using the least squares method, we obtained the following linear adjustment

$$f(n) = 0.2667n + 0.6182 \approx \frac{n}{3.7} + 0.6182$$

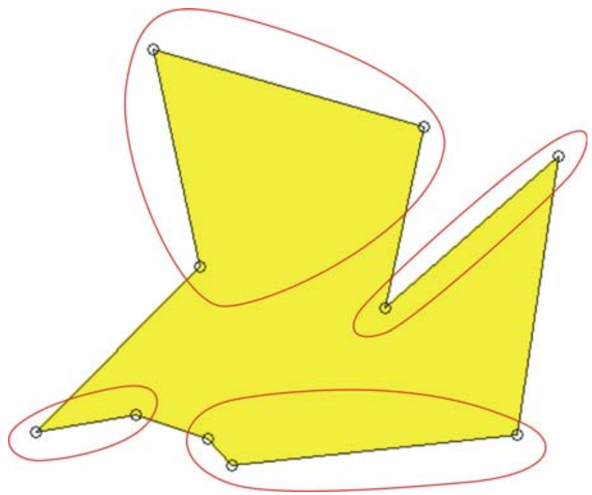
On average, the maximum number of hidden vertices in an arbitrary polygon P with n vertices is $\left\lfloor \frac{n}{3.7} \right\rfloor$

Approximation Ratio

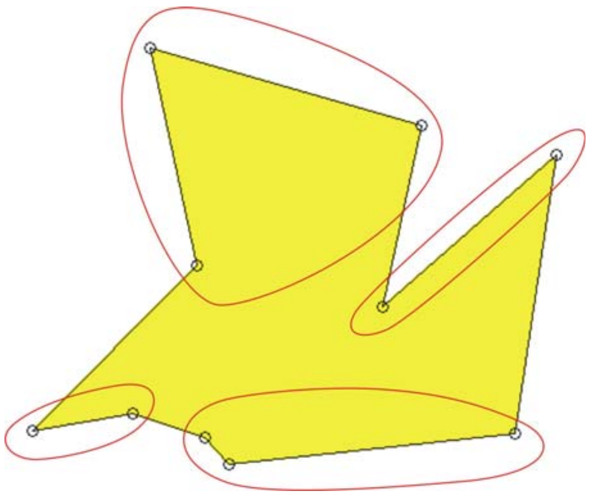
How can to prove that our approximate solutions are "good"?

UPPER BOUND for the optimal solution of the problem

CLIQUE PARTITION of **VG(P)**



Approximation Ratio



For each clique of the partition **C** we can hide at most one vertex in **P**

$$|C| \geq |H| \qquad \forall\, C, H$$

$$|C| \geq a(P) \geq h(P) \geq |H|$$

$a(P)$ number of cliques in a minimum-cardinality clique partition

$h(P)$ number of hidden vertices in a maximum-cardinality hidden vertex set

Approximation ratio of the solution **H*** $\qquad h(P) \,/\, |H^*|$

We obtain a hidden vertex set **H*** that approximates $h(P)$ with approximation ratio $|C| / |H^*|$

Approximation Ratio

- The problem of determining *a*(P) (Minimum Clique Partition problem) is ***NP*-Hard**, so we developed a greedy algorithm to obtain one solution **C**

- M_1 algorithm has an approximation ratio of 1.7, being equal to 3/2 for 98.44% of the instances

- This means that, the obtained approximate solution, **|H*|**, has
  - at least 1/1.7 of the optimal number of hidden vertices, for all instances;
  - And at least 2/3 of the optimal number of hidden vertices, for 98.44% of the instances

Experiments & Results: Orthogonal Polygons

- A similar study was made for orthogonal polygons

  - The best algorithm is M_1 (case: $T_0 = n$ and FSA decrease), with significantly different results from the other algorithms
  - On average the maximum number of hidden vertices in an orthogonal polygon P with n vertices is $n/4$
  - The approximation ratio is $3/2$, for all randomly generated instances

Future Work

- Different parametrizations of genetic algorithms
- Hybrid metaheuristics

PROBLEMS

- ➢ Optimal triangulations, polyhedral terrains
- ➢ Optimal polygonizations, watchman routes
- ➢ Cooperative guarding, another variants, ...
- ➢ Rectangular partitions, ...

	COMBINATORIAL BOUND	APPROX ALGORITHM
VERTEX-GUARD	n/3	n/6.48
HIDDEN VERTEX	n/2	n/3.7
2-MODEM ILLUMINATION	¿n/6?	n/26
4-MODEM ILLUMINATION	¿?	n/52

Hiding Points in Polygons using Approximation Algorithms

Gracias por su atención

Gregorio Hernández
Universidad Politécnica de Madrid