UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Câmpus Curitiba (PR)

LABIC

Laboratório de Bioinformática e Inteligência Computacional - UTFPR

http://labic.utfpr.edu.br

Computação Evolucionária

Prof. Heitor Silvério Lopes

hslopes@utfpr.edu.br

http://silverio.net.br/heitor

class6b (2020)

Programação Genética

- Parte I - Fundamentos:
  - Origens, literatura, contextualização, resolução de problemas, indução de programas
  - Conjuntos de funções e terminais, geração da população inicial, medidas de *fitness*, operadores principais e secundários, critério de parada
- Parte IIa – Aplicações:
  - Exemplos de aplicação clássicos (cap. 7):
    - Regressão simbólica
    - Formiga artificial
- Parte IIb – Aplicações:
  - Evolução de comportamentos emergentes (cap. 12):
    - Busca de alimento
    - Priorização de tarefas
  - Evolução da classificação (cap. 17):
    - Mineração de dados
    - Reconhecimento de padrões
  - Evolução de estratégias (cap. 15):
    - Estratégia de jogo perseguidor-evasor
  - Software Lil-GP

Evolução de Comportamentos Emergentes

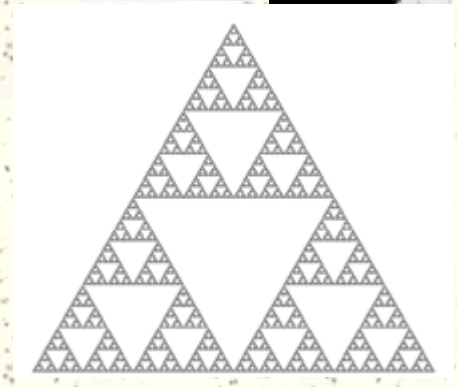
(cap. 12)

Evolução de comportamentos emergentes

- A aplicação repetitiva de regras simples pode levar a comportamentos complexos

Importante !

- Comportamentos emergentes aparecem em:
  - Fractais e sistemas caóticos
  - *L-systems*, e Autômatos celulares
  - Redes neurais
  - *Swarm intelligence*, p. ex. ACO e PSO
  - Na natureza !!



Evolução de comportamentos emergentes

- Proposta: utilizar Programação Genética para evoluir regras simples, tendo como objetivos:
  - Gerar comportamentos complexos ao aplicar repetidamente as regras simples
  - Imitar a natureza

- Dois casos distintos (*toy problems*):
  - Estudo de caso #1: Busca por alimento (imitação de insetos sociais)
  - Estudo de caso #2: Priorização de tarefas (imitação de tomada de decisão sob diferentes condições)

Estudo de caso #1: Busca por Alimento

- Objetivo:
  - Criar um conjunto de regras que, quando executado simultaneamente pelos indivíduos de um grupo, causa o surgimento de um comportamento coletivo benéfico de alto nível.
  - Observar a influência do comportamento de cada indivíduo na comunidade e vice-versa
- Contexto:
  - Evoluir um programa que faça com que formigas artificiais exibam um comportamento semelhante à natureza para transportar para o ninho o alimento disponível para coleta.

Busca por alimento (ou de outro recurso)

- Contexto do problema: Formigas artificiais devem explorar cooperativamente a vizinhança do ninho em busca de alimento e coletá-los
- O comportamento coletivo deve mudar ao longo do tempo de acordo com a concentração/dispersão do recurso (comida):
  - Quando estiver concentrado em certas regiões:
    - É vantajoso concentrar a busca nas regiões promissoras
  - Quando estiver disperso:
    - Sempre é melhor cada indivíduo buscar aleatoriamente
  - Porém, inicialmente não se sabe onde se concentra o alimento

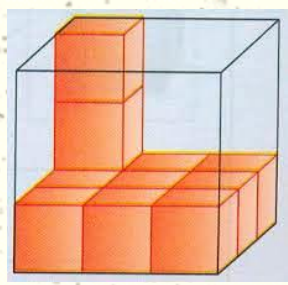
Formigas !!??

- Se comunicam indiretamente:
  - Usam trilhas de feromônio

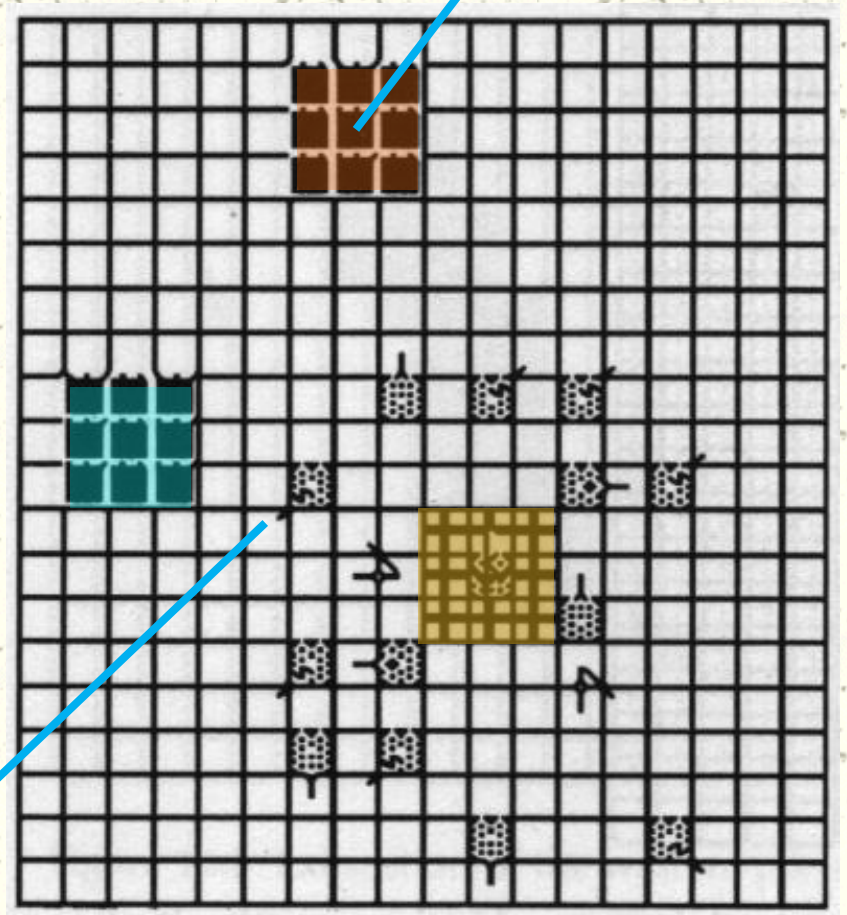
- Não há coordenação central no formigueiro
  - Cada formiga tem um conjunto de regras internas simples
  - A formiga é reativa em relação ao ambiente

- Aqui <u>não há</u> relação com o ACO

Busca por alimento: Simulação computacional



- Descrição do ambiente
  - Grade toroidal de 32x32
  - Duas pilhas (norte e oeste) de 3x3x8=144 alimentos
- Inicialização
  - 20 formigas no ninho com direção aleatória
  - Cada formiga: 8 direções + indicador de carga de alimento
- Um <u>único</u> programa para <u>todas</u> as formigas



Parâmetros da PG

- Objetivo: Trazer os alimentos para o ninho
- Terminais:
  - MOVE-RANDOM: escolhe uma direção e move-se dois passos
  - MOVE-TO-NEST: move-se um passo em direção ao ninho
  - PICK-UP: pega o alimento da posição corrente (se houver e se já não estiver carregando alimento)
  - DROP-PHEROMONE: se estiver carregando alimento, deposita feromônio na posição corrente que se propaga para as 3x3 posições vizinhas
- Funções: *F* ={IF-FOOD-HERE, IF-CARRING-FOOD, MOVE-TO-ADJACENT-FOOD-ELSE, MOVE-TO-ADJACENT-PHEROMONE-ELSE, PROGN}

  aridades: 2, 2, 1, 1, 2

Parâmetros da PG

- Casos de *fitness*: 1 único labirinto
- *Fitness* bruto: número de alimentos transportados para o ninho num período de tempo preestabelecido
- Fitness padronizado: 144-r
- M=500, G=51
- Tempo máximo: 400 passos cada execução
- Observações:
  - Todas as formigas executam uma única ação por instante de tempo. O efeito de cada ação influencia o ambiente e as demais formigas
  - Mais de uma formiga pode ocupar o mesmo ponto no labirinto

Busca por alimento - solução

- Solução com 100% de *fitness* obtida em 9 gerações (≈4500 avaliações)

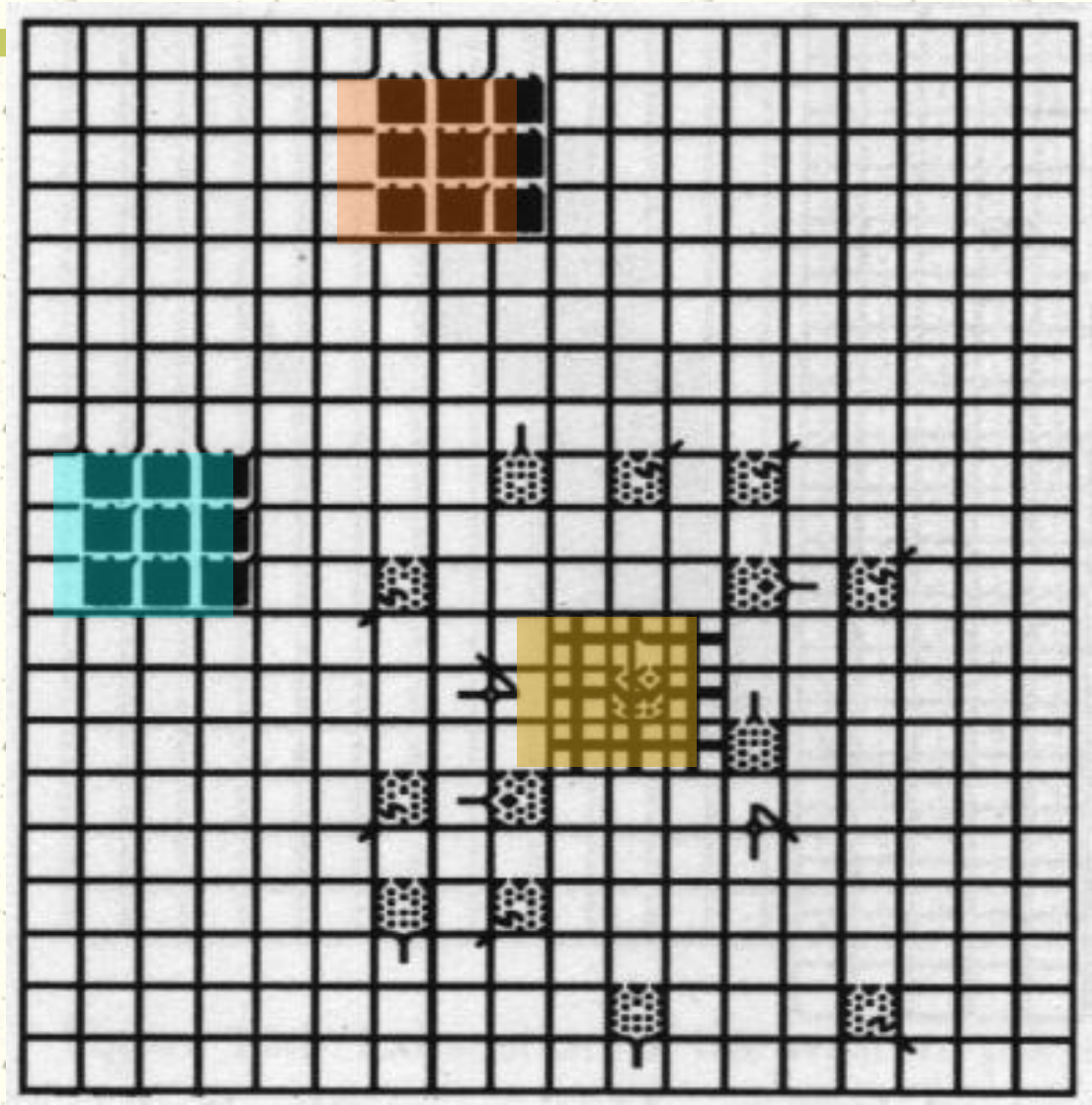
```
(PROGN (PICK-UP) (IF-CARRYING-FOOD (PROGN (MOVE-TO-
ADJACENT-PHEROMONE-ELSE (MOVE-TO-ADJACENT-FOOD-ELSE
(MOVE-TO-ADJACENT-FOOD-ELSE (MOVE-TO-ADJACENT-FOOD-ELSE
(PICK-UP)))))) (PROGN (PROGN (PROGN (PROGN (MOVE-TO-
ADJACENT-FOOD-ELSE (PICK-UP)) (PICK-UP)) (PROGN (MOVE-TO-
NEST) (DROP-PHEROMONE))) (PICK-UP)) (PROGN (MOVE-TO-NEST)
(DROP-PHEROMONE)))) (MOVE-TO-ADJACENT-FOOD-ELSE (IF-
CARRYING-FOOD (PROGN (PROGN (DROP-PHEROMONE) (MOVE-TO-
ADJACENT-PHEROMONE-ELSE (IF-CARRYING-FOOD (MOVE-TO-
ADJACENT-FOOD-ELSE (PICK-UP)) (MOVE-TO-ADJACENT-FOOD-ELSE
(PICK-UP))))) (MOVE-TO-NEST)) (IF-FOOD-HERE (PICK-UP)
(IF-CARRYING-FOOD (PROGN (IF-FOOD-HERE (MOVE-RANDOM) (IF-
CARRYING-FOOD (MOVE-RANDOM) (PICK-UP))) (DROP-PHEROMONE))
(MOVE-TO-ADJACENT-PHEROMONE-ELSE (MOVE-RANDOM)))))))).
```

Busca por alimento - programa

```
1  (PROGN (PICK-UP)
2         (IF-CARRYING-FOOD
3           (PROGN (MOVE-TO-ADJACENT-PHEROMONE-ELSE
4                    (MOVE-TO-ADJACENT-FOOD-ELSE (PICK-UP)))
5                  (MOVE-TO-ADJACENT-FOOD-ELSE (PICK-UP))
6                  (MOVE-TO-NEST)
7                  (DROP-PHEROMONE)
8                  (MOVE-TO-NEST)
9                  (DROP-PHEROMONE))
10          (MOVE-TO-ADJACENT-FOOD-ELSE
11            (IF-FOOD-HERE
12              (PICK-UP)
13              (MOVE-TO-ADJACENT-PHEROMONE-ELSE
14                (MOVE-RANDOM))))))).
```

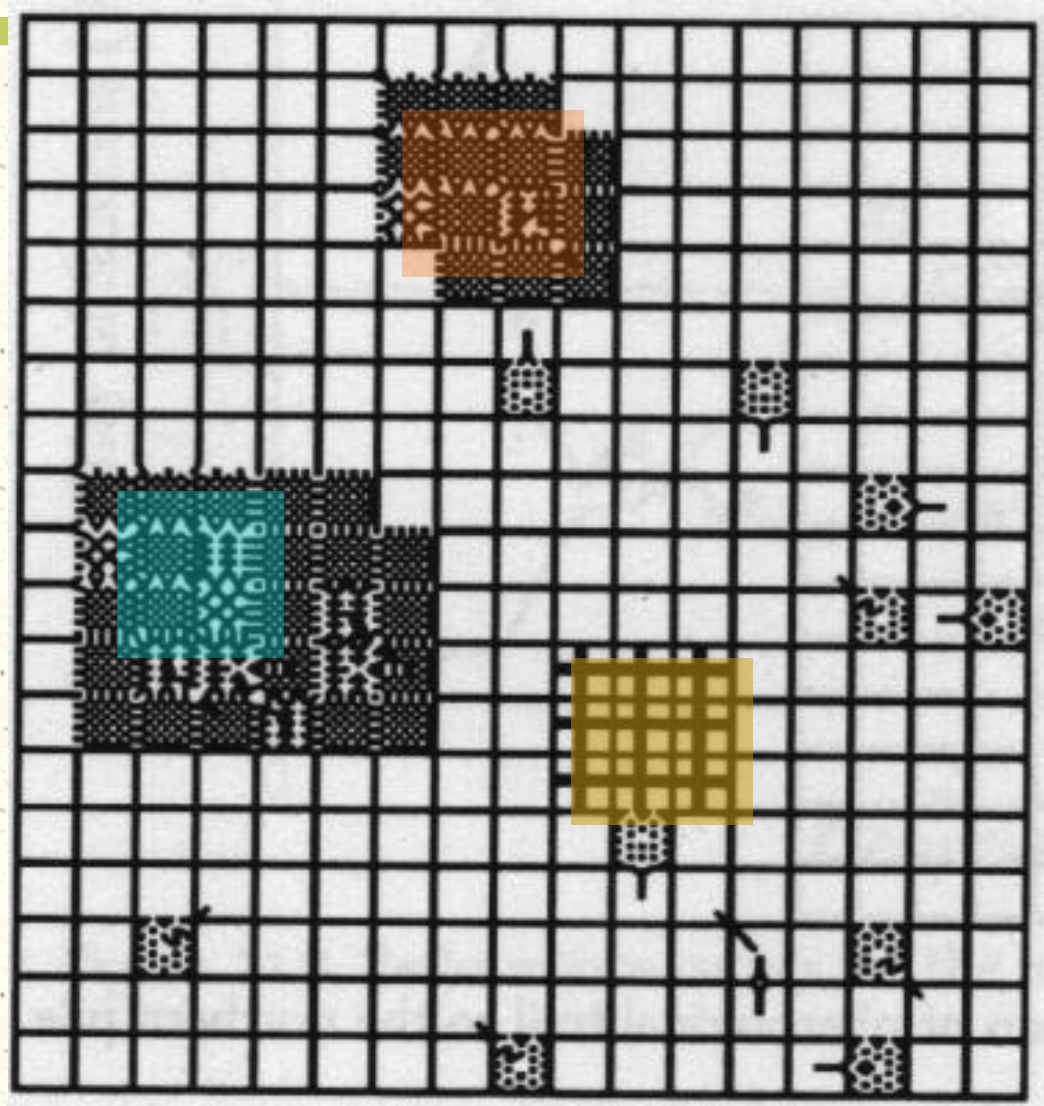
Simulação - Busca por alimento

- Fase 1: busca aleatória



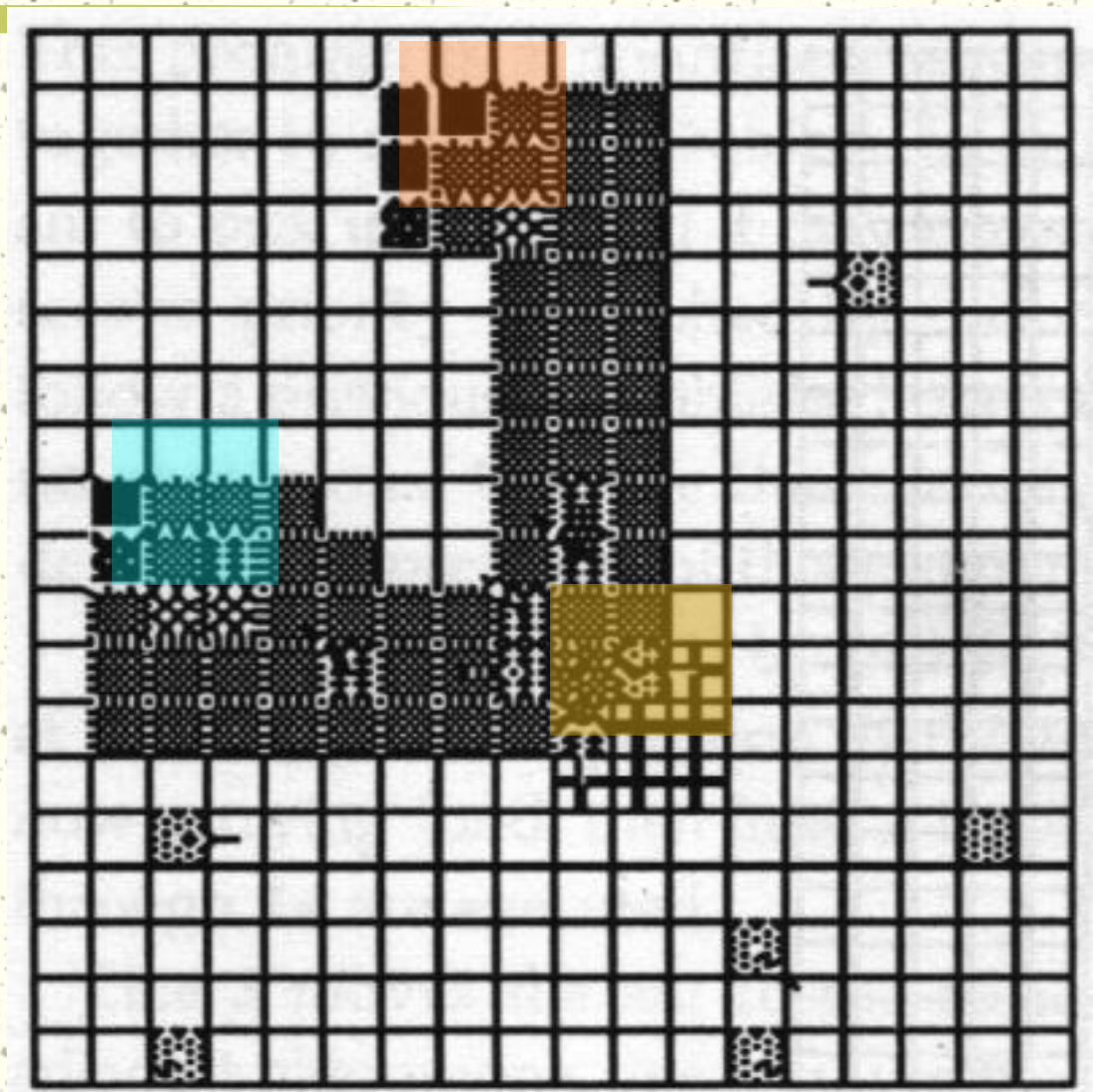
Simulação - Busca por alimento

- Fase 2: contato inicial com o alimento e formação das trilhas de feromônio



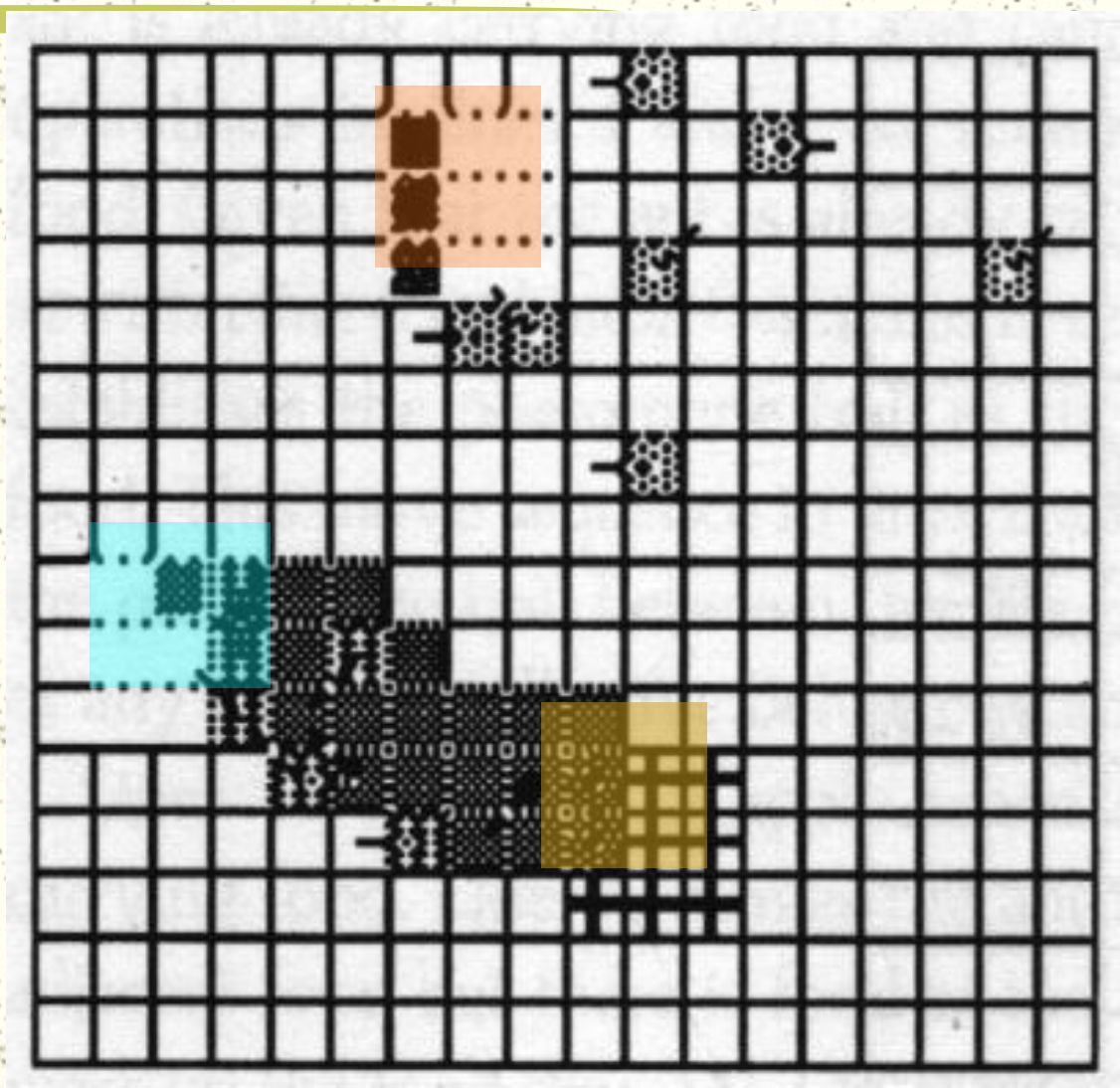
Simulação - Busca por alimento

- Fase 3: estabelecimento de trilhas de feromônio entre as pilhas de alimento e o ninho



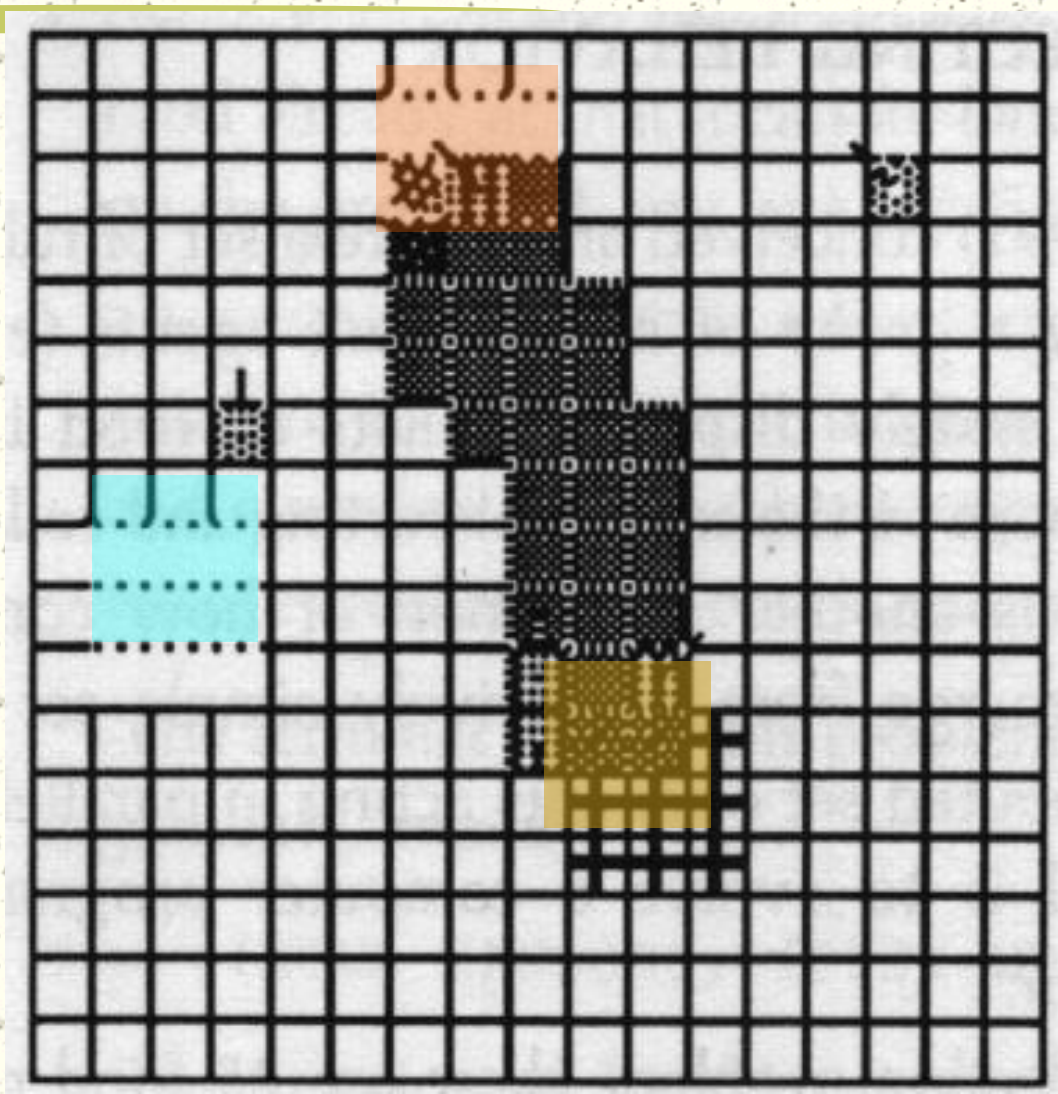
Simulação - Busca por alimento

- Fase 4: Desintegração prematura da trilha de feromônio da pilha norte



Simulação - Busca por alimento

- Fase 5: Exaustão da pilha leste e exploração final da pilha norte

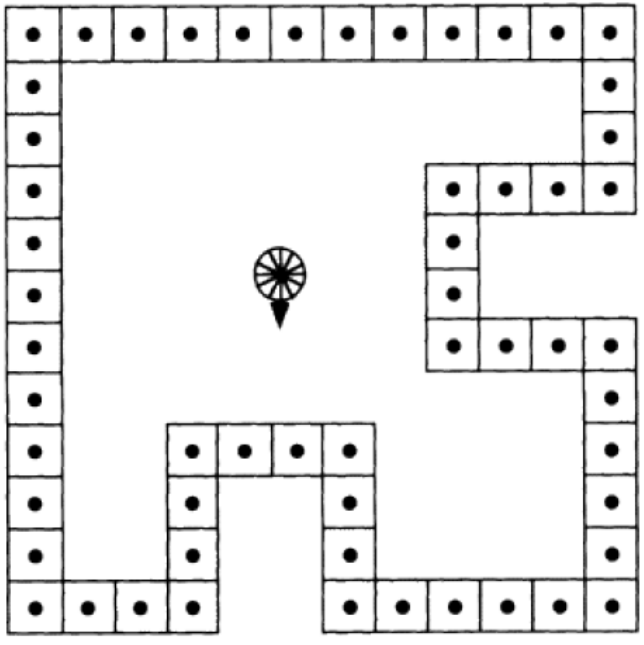


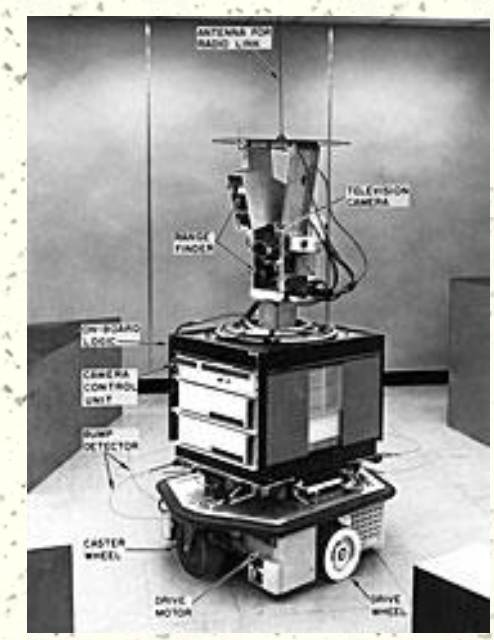
Aplicabilidade: navegação de robôs móveis

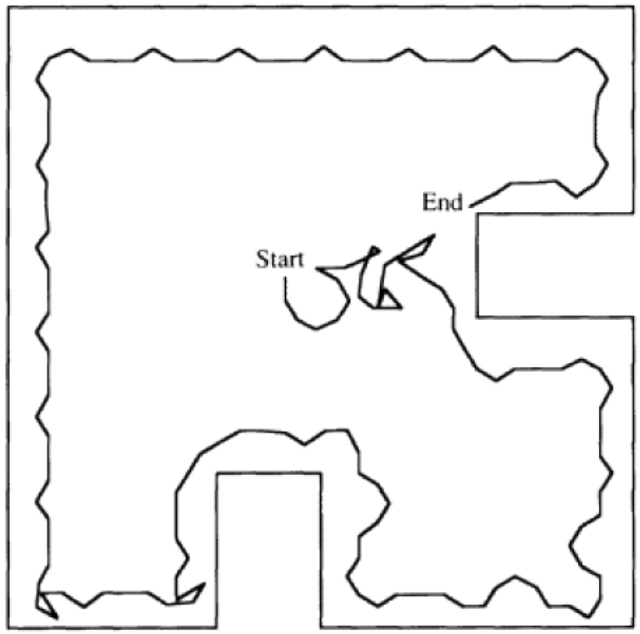
- Exploração espacial (*Mars exploration roover*) buscam pedras na superfície do planeta e trazem de volta à nave
- Aspirador de pó autônomo
- Robôs militares: para reconhecimento, ataque ou resgate

Problema prático equivalente: Navegação de robôs móveis

- Objetivo: encontrar um programa que, quando executado pelo robô, lhe permita navegar pelo ambiente seguindo a parede, sem colisões.







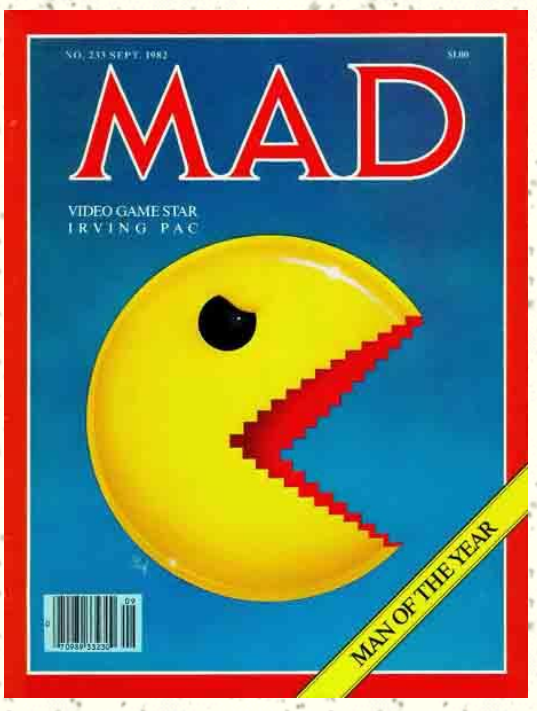
Estudo de caso #2: Priorização de Tarefas

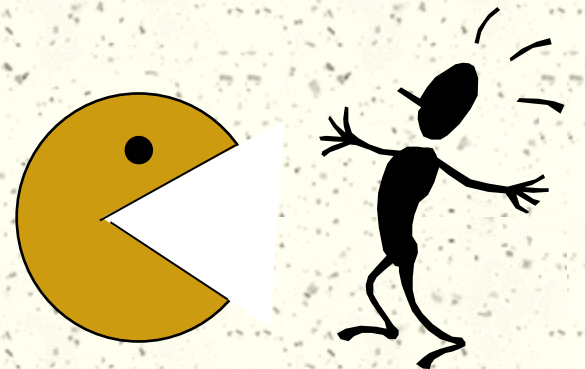
- O problema de planejamento de tarefas envolve estabelecer prioridades entre tarefas com importâncias e urgências distintas.

- Quanto um evento extraordinário ocorre, um arranjo diferente de prioridades é requerido

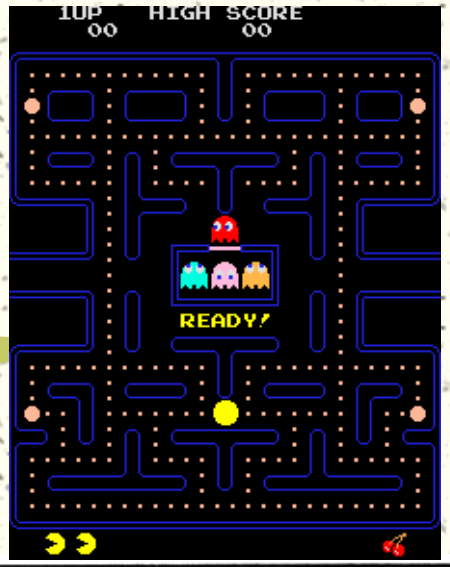
O Pac Man

- *Pac Man*: videogame da década de 80
- Objetivo: maximizar os pontos ganhos
- Os monstros caçam o *Pac Man*, mas podem ser imprevisíveis. Também "revivem" após certo tempo
- As prioridades mudam ao longo do jogo que termina quando não há mais sabão para comer

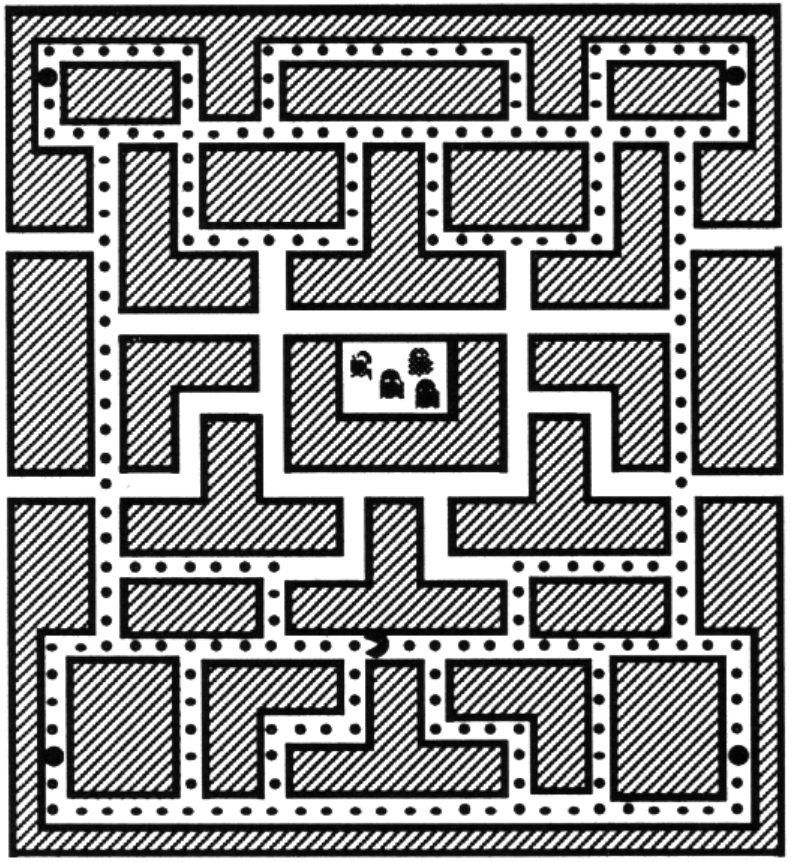




Pac Man – condições iniciais



- *Grid* toroidal de 31 x 28.
- O movimento é restrito a corredores
- Dois túneis conectam ambos os lados
- O *Pac Man* começa na posição (13,23)
- 4 monstros iniciam na posição (13,11)



Escala de prioridades

- No estado inicial (normal):
  - 1- fugir dos monstros
  - 2- perseguir e capturar a frutinha móvel
  - 3- comer sabão
- Quando o *Pac Man* come um sabão mágico, os papéis mudam entre *Pac Man* e os monstros, assim as prioridades ficam:
  - 1- comer os monstros (azuis)
  - 2- não comer sabão mágico

Pac-Man - Modelagem

- Terminais: (*T*)
  - APILL, RPILL, DISPILL (sabão mágico)
  - AGA, RGA, DISGA (monstro A)
  - AGB, RGB, DISGB (monstro B)
  - AFOOD, DISU (Comida)
  - AFRUIT, DISF (frutinha móvel)
- Conjunto de Funções (*F*)
  - IFB (se azul a caso contrário b)
  - IFLTE (se a ≤ b então c senão d)

Pac Man – parâmetros da PG

- *Fitness* cru: Pontos do jogo
- *Fitness* padronizado: 18220-r
- Casos de *fitness*: 1 rodada
- M=500, G=51

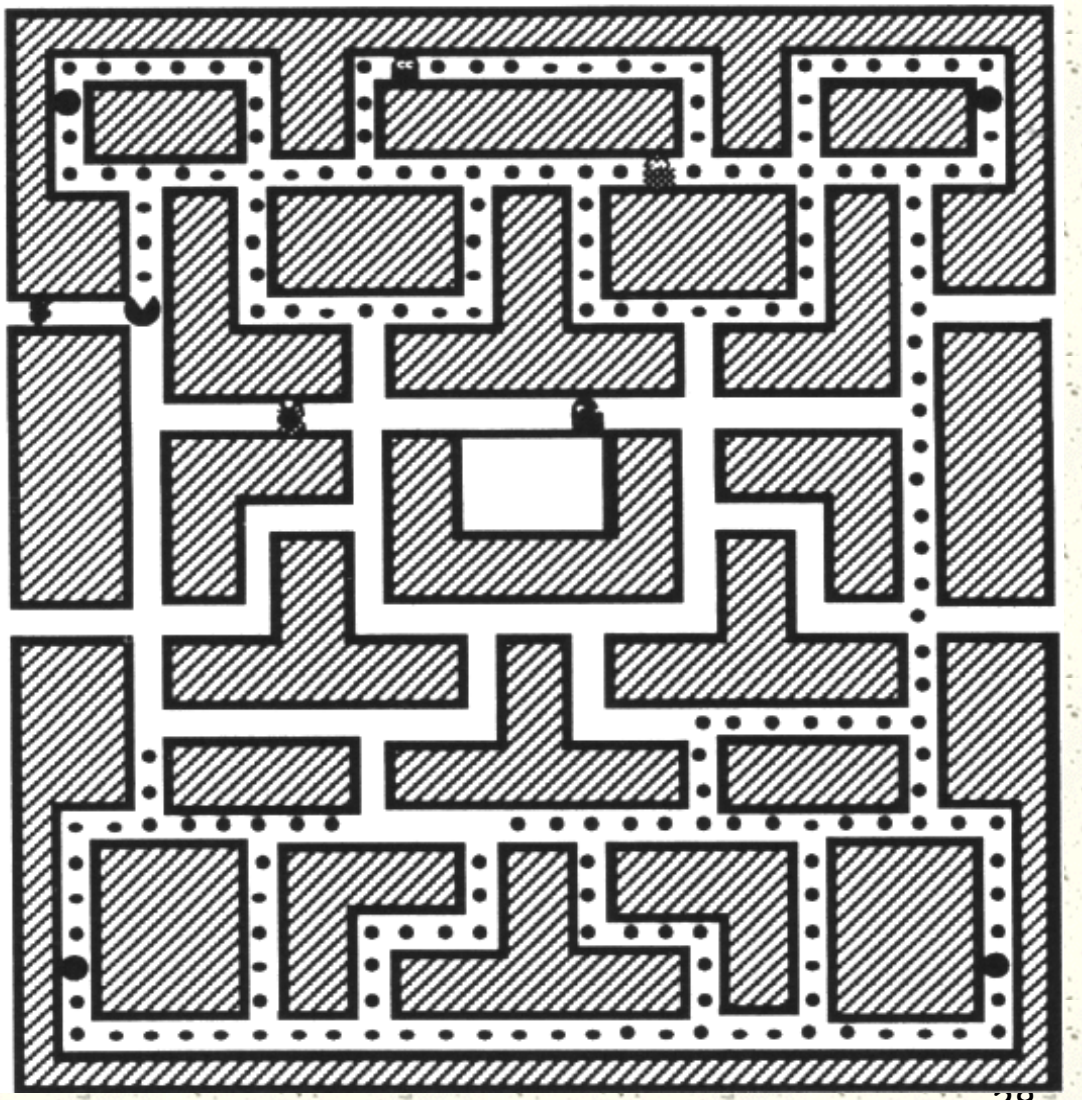
Pac Man – programa

- Melhor solução obtida na geração 35

```
(IFB (IFB (IFLTE (AFRUIT) (AFRUIT) (IFB (IFB (IFLTE
(IFLTE (AGA) (DISGA) (IFB (IFLTE (DISF) (AGA) (DISPILL)
(IFLTE (DISU) (AGA) (AGA) (IFLTE (AFRUIT) (DISU) (AFRUIT)
(DISGA)))) (IFLTE (AFRUIT) (RGA) (IFB (DISGA) 0)
(DISGA))) (DISPILL)) (IFB (IFB (AGA) (IFLTE (IFLTE (IFLTE
(AFRUIT) (AFOOD) (DISGA) (DISGA)) (AFRUIT) 0 (IFB (AGA)
0)) (DISPILL) (IFLTE (AFRUIT) (DISPILL) (RGA) (DISF))
(AFRUIT))) 0) (AGA) (RGA)) (AFRUIT)) (IFLTE (IFLTE (RGA)
(AFRUIT) (AFOOD) (AFOOD)) (IFB (DISPILL) (IFLTE (RGA)
(APILL) (AFOOD) (DISU))) (IFLTE (IFLTE (RGA) (AFRUIT)
(AFOOD) (RPILL)) (IFB (AGA) (DISGB)) (IFB (AFOOD) 2) (IFB
(DISGB) (AFOOD))) (IFB (DISPILL) (AFOOD)))) (RPILL)) (IFB
(DISGB) (IFLTE (DISU) 0 (AFOOD) (AGA)))) (IFB (DISU)
(IFLTE (DISU) (DISU) (IFLTE (IFLTE (AFRUIT) (AFOOD)
(DISPILL) (DISGA)) (AFRUIT) 0 (IFB (AGA) 0)) (RGB)))).
```

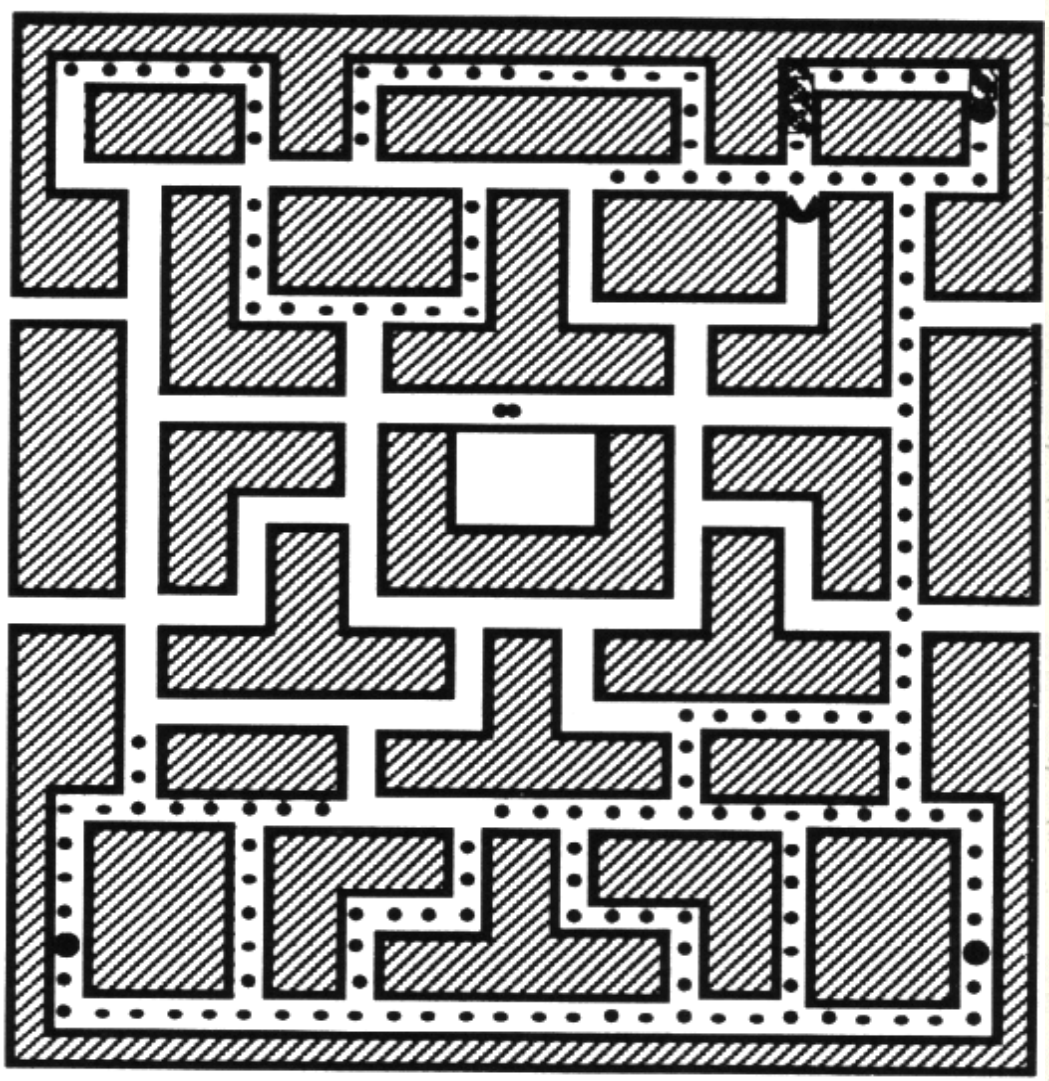
Pac Man – simulação

- Tempo 25, antes de comer a frutinha mágica do túnel oeste



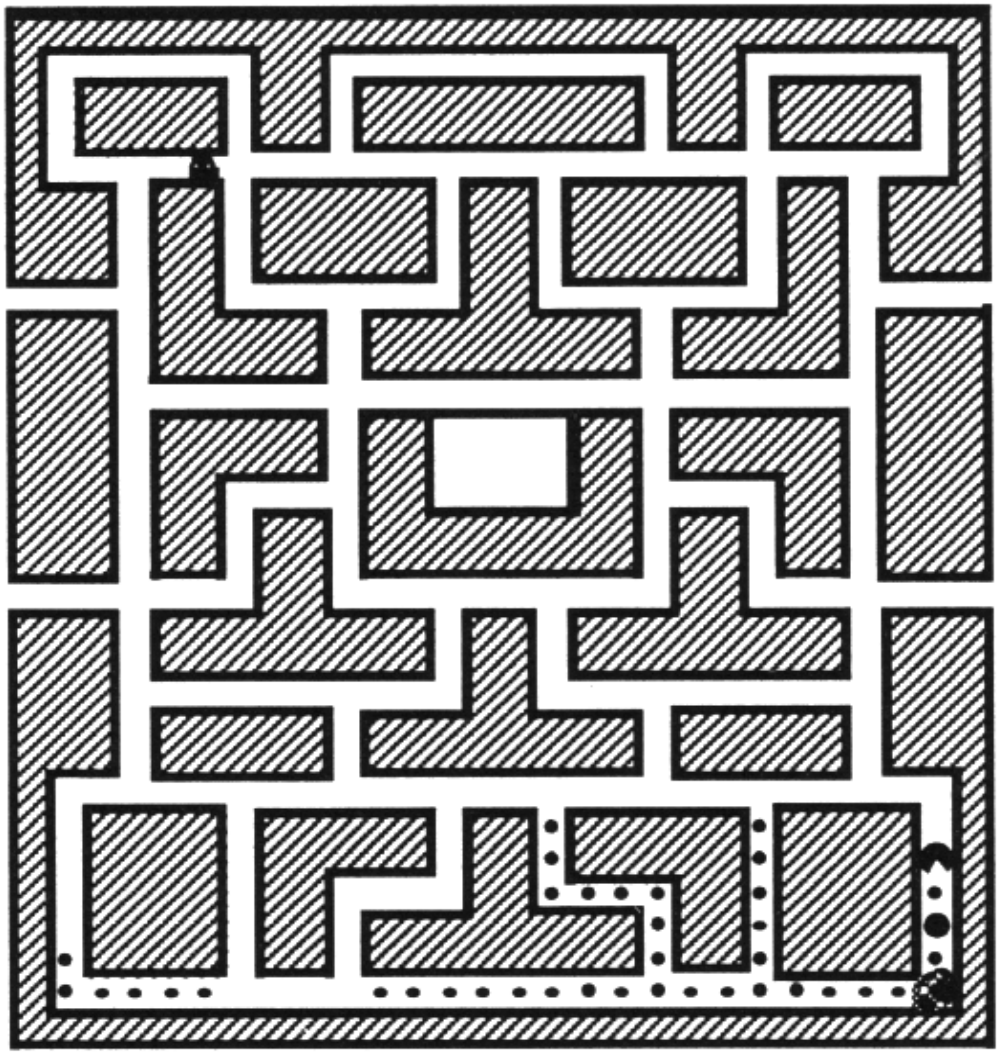
Pac Man – simulação

- Tempo 66, perseguindo os monstros



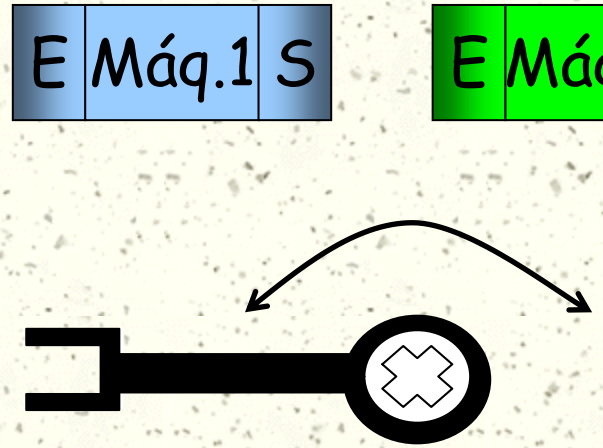
Pac Man – simulação

- Tempo 301, perseguindo três monstros.
- No final (tempo 405), o programa obteve 9220 pontos de um total de 18220



Aplicação prática: robô de linha de montagem

- Elementos:
  - Uma esteira de entrada de peças brutas
  - Uma esteira de saída de peças produzidas
  - Duas máquinas-ferramenta com depósitos restritos de entrada e saída
  - Braço robótico capaz de pegar e transportar uma peça de cada vez entre cada equipamento
- Problema:
  - Encontrar um algoritmo de controle do robô para que se produza o maior número possível de peças por hora



Evolução da Classificação

(cap. 17)

Evolução da Classificação

- Há muitas abordagens diferentes para criar um classificador de dados: árvores de decisão, conjunto de regras, redes neurais, classificadores Bayesianos, etc
- A representação natural em PG é uma árvore
- Isto sugere que PG pode ser útil para evoluir uma árvore de decisão para classificar dados
- Há uma relação direta entre árvores de decisão e regras de classificação

Árvores de decisão

- Métodos para indução de árvores de decisão: C4.5, ID3
- ID3:
  - Classificador hierárquico criado por Quinlan (1986), capaz de associar objetos a classes
  - Cada objeto pertence somente a uma classe e é descrito por um conjunto de atributos
  - Após treinado, o ID3 pode classificar objetos não apresentados na fase de treinamento.
  - Nós externos (nó folha) representam classes
  - Nós internos testam atributos

Evolução de uma árvore de decisão com PG

- Objetivo: encontrar uma A.D. para analisar o tempo e decidir se vai passear ou não

- **Atributos:** Outlook, Temperature, Humidity, Windy
- **Valores possíveis dos atributos:**
  - Temperatura: {hot, normal, cool}
  - Tempo: {sunny, overcast, rainy}
  - Vento: {sim, não}
  - Umidade: {high, normal}
  - **Classes:** 0 e 1

Inst.	Outlook	Temp	Humidity	Windy	Play
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Funcionamento da árvore de decisão

- Quando um objeto é apresentado à árvore de decisão, a função raiz testa os atributos deste objeto, e executa o argumento resultante do teste
- Se o argumento executado encontrou um terminal, um nome de classe é retornado
- Se o argumento executado é outra função de teste, outro atributo é testado, e assim por diante até chegar num terminal e o nome de uma classe seja retornado

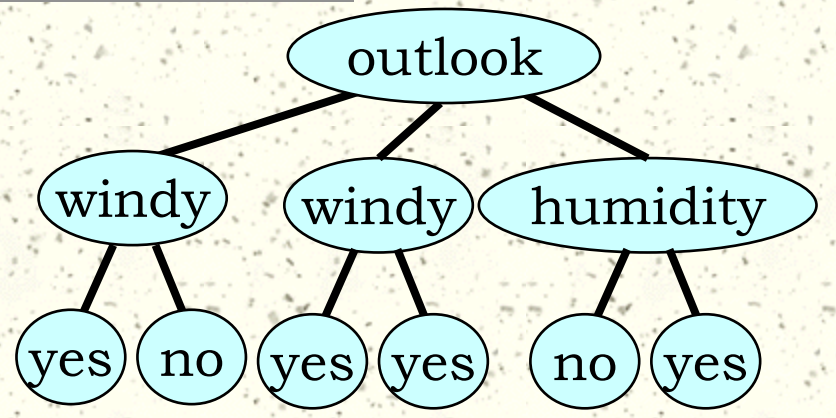
Evolução de uma árvore de decisão com PG

- Dados de entrada:
  - Precisam ser normalizados → respeitar a propriedade do Fechamento
- Conjuntos:
  - $\mathcal{F}$={outlook,temp,hum,wind}
  - $\mathcal{T}$={0,1}
- Casos de fitness: 14 exemplos
- Fitness:
  - Fitness cru é o número de acertos entre o exemplo dado e o valor predito pela árvore de decisão
  - Fitness padronizado = 14 – fitness cru
- M=500, G=51

Evolução de uma árvore de decisão com PG

- Melhor resultado encontrado na geração 8 (*fitness* máximo)
- S-expression:

```
(OUTLOOK (WINDY 1 0)
         (WINDY 1 1)
         (HUMIDITY 0 1))
```



- Principal problema: A PG não consegue simplificar as árvores (árvores não podadas)

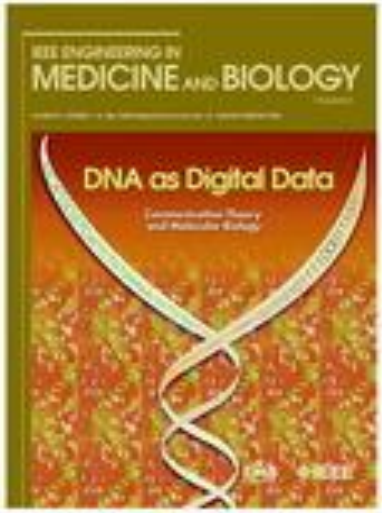
Montagem de uma árvore de decisão

- Passos preparatórios:
  - Definir o conjunto de classes (nós folha)
  - Definir o conjunto de funções que testam os atributos (nós internos)
  - Definir uma medida de *fitness*
  - Definir os exemplos de treinamento
  - Definir o valor dos parâmetros numéricos que controlam o crescimento da árvore
  - Definir o critério de término

Aplicação prática 1: Mineração de dados

- Principal tarefa: descoberta de regras de classificação
- Alternativamente pode ser utilizado para análise associativa (regras de associação)
- Como efeito colateral, realiza também uma espécie de seleção de atributos
- Cada árvore pode ser uma regra simples (abordagem de Michigan) ou um conjunto de regras (abordagem de Pittsburgh)
- Em geral é preciso utilizar algum tipo de restrição sintática (*constrained syntax/strongly typed*)

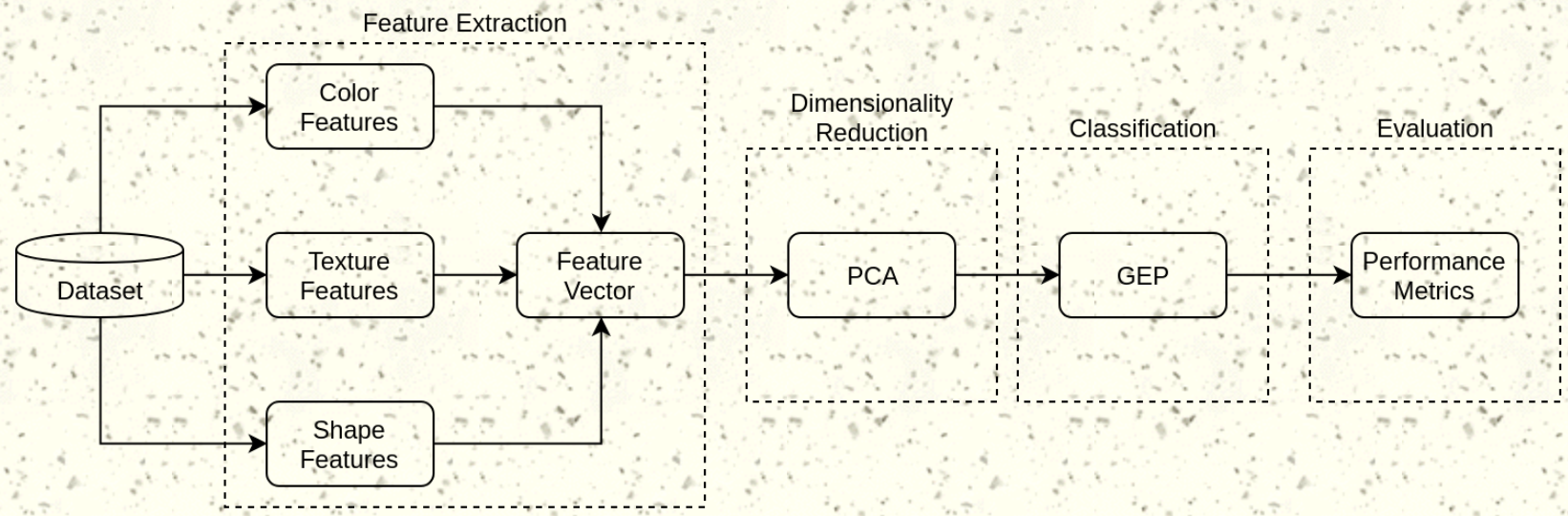
Aplicação prática 1: Mineração de dados



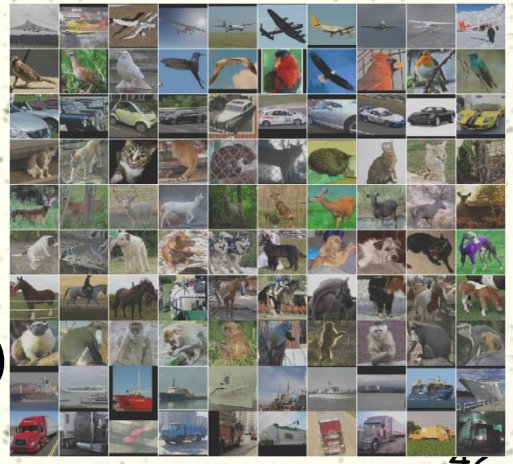
- Busca de regras para diagnóstico diferencial de 12 patologias relacionadas a dor torácica
  - Bojarczuk, C.C., Lopes, H.S., Freitas, A.A. Genetic programming for knowledge discovery in chest pain diagnosis. ***IEEE Engineering in Medicine and Biology Magazine***, v. 19, n. 4, p. 38-44, july/august, 2000.

- Busca de regras para a previsão de reincidência do câncer de mama e para previsão de sobrevida para pacientes com câncer infantil
  - Bojarczuk, C.C., Lopes, H.S., Freitas, A.A. A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets. ***Artificial Intelligence in Medicine***, v. 30, n. 1, p. 27-48, 2004.

Aplicação prática 2: classificação multiclasse de imagens



- Imagens são transformadas em um vetor de características (*features*) compostos de 2592 elementos:
  - Cor: Histograma de cor – HOG (768)
  - Textura: Padrões binários lineares - LBP (256)
  - Forma: Moments de Hu (1568)



Aplicação prática 2: classificação multiclasse de imagens




- *Datasets*:
  - Caltech: 5 classes: aviões, fundos neutros, faces, motos, relógios
  - STL-100: 10 classes: aviões, pássaros, carros, gatos, veados, cachorros, cavalos, macacos, navios e caminhões

Aplicação prática 2: classificação multiclasse de imagens

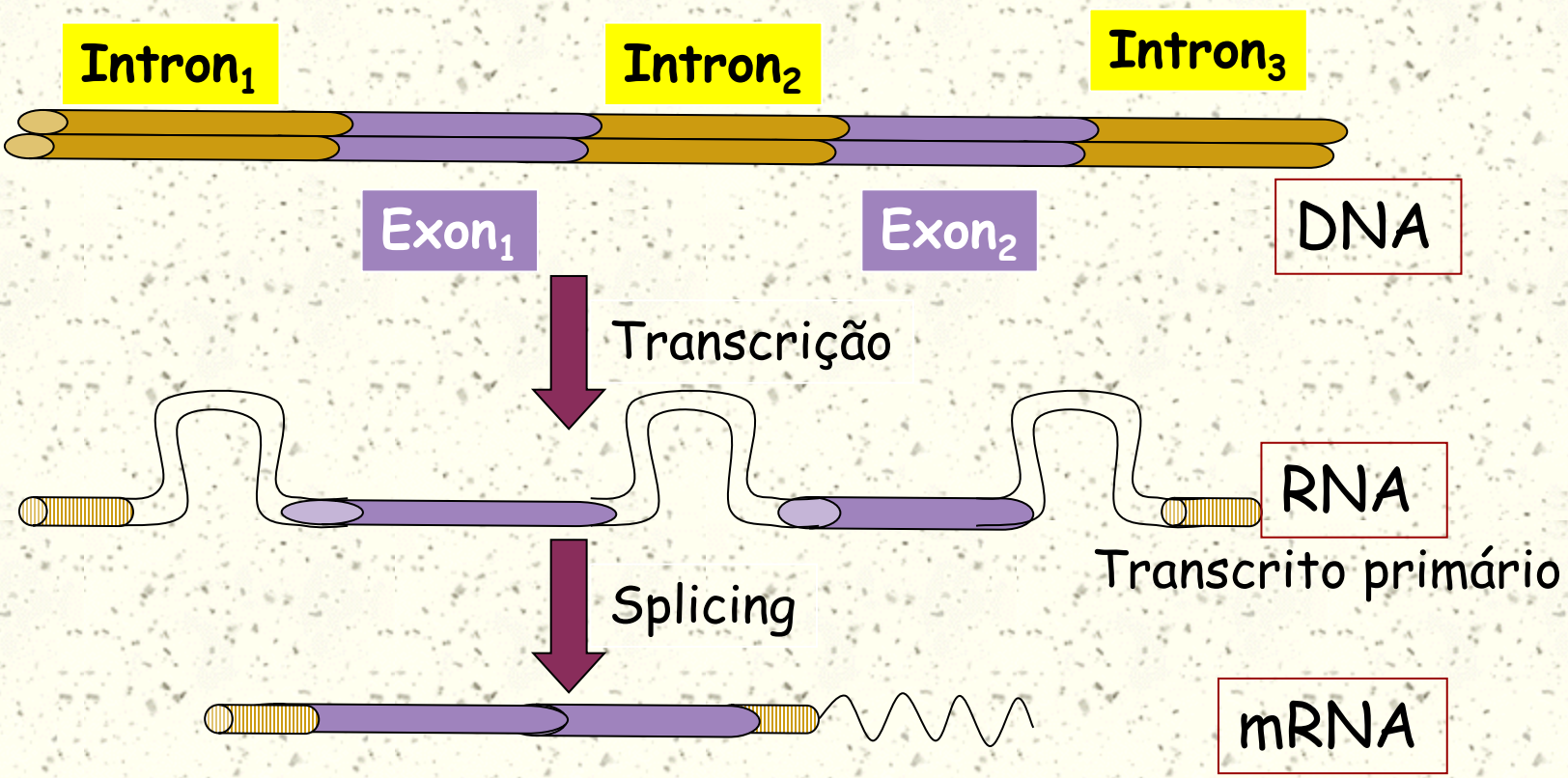
- Várias medidas de *fitness* foram testadas
- Os resultados da classificação foram melhores ou similares ao *baseline* C4.5 e *Random Forest*
- PG obteve regras mais simples do que os outros métodos (compreensibilidade é importante !)

Romero Aquino, N.M, Ribeiro, M., Gutoski, G., Benítez, C.M.V., Lopes, H.S. A Gene Expression Programming Approach for Evolving Multi-Class Image Classifiers. ***In: Proc. IEEE Latin-American Conference on Computational Intelligence***, 2017

PG para reconhecimento de padrões

- Problema:
  - Identificação de pseudo-exons numa sequência de DNA
  - DNA é formado por 4 bases $\{A,C,G,T\}^n$
  - O DNA humano tem cerca de 3 bilhões de bases, só 1-2% codificam proteínas.
- Através de PG busca-se uma expressão capaz de identificar uma sequência de nucleotídeos com 5 caracteres de comprimento, em uma sequência de 1000 caracteres.
  - O pseudo-exon tem a forma AA*TT
  - Exemplo real: um intron sempre inicia com GT e termina com AG

Mapeamento DNA → mRNA



Parâmetros da PG

- *T* ={(AIF),(CIF),(GIF),(TIF),(MHG)}
  - AIF retorna T(*true*) se uma Adenina for encontrada, caso contrário, NIL
  - CIF, GIF, TIF, idem, para Citosina, Guanina, Timina
  - MHG avança uma posição à direita na sequência e retorna T(*true*).
- *F* ={AND, OR, NOT}, M=500, *G*=51
- Casos de *fitness*: 1000 bases da sequência sob teste. *Fitness* bruto: distância de Hamming entre o valor retornado pela expressão e a classificação correta para aquela posição

Resultados

- O melhor indivíduo da geração 35 obteve um *fitness* de 0:

```
(AND (AND (OR (AND (AIF) (AIF)) (AND (AIF) (AIF)))
(OR (MHG) (OR (OR (OR (OR (GIF) (NOT (OR (MHG)
(GIF)))) (GIF)) (AND (NOT (GIF)) (NOT (OR (NOT
(AIF)) (MHG))))) (OR (AIF) (NOT (OR (AIF) (AND (GIF)
(AIF)))))))) (AND (TIF) (OR (NOT (NOT (TIF))) (AND
(NOT (MHG)) (NOT (NOT (OR (OR (NOT (TIF)) (GIF)) (OR
(NOT (AIF)) (NOT (OR (AIF) (AIF)))))))))))
```

- Trecho do DNA identificando os pseudo-exons:

```
GGTAAAACTTTTTCTGGGTTCCAATTTTTCGTCACTAGTTGTAAAGCTGA
00000111110000000000011111000000000000000000000
```

Aplicação prática: detecção de *motifs* para classificação de proteínas

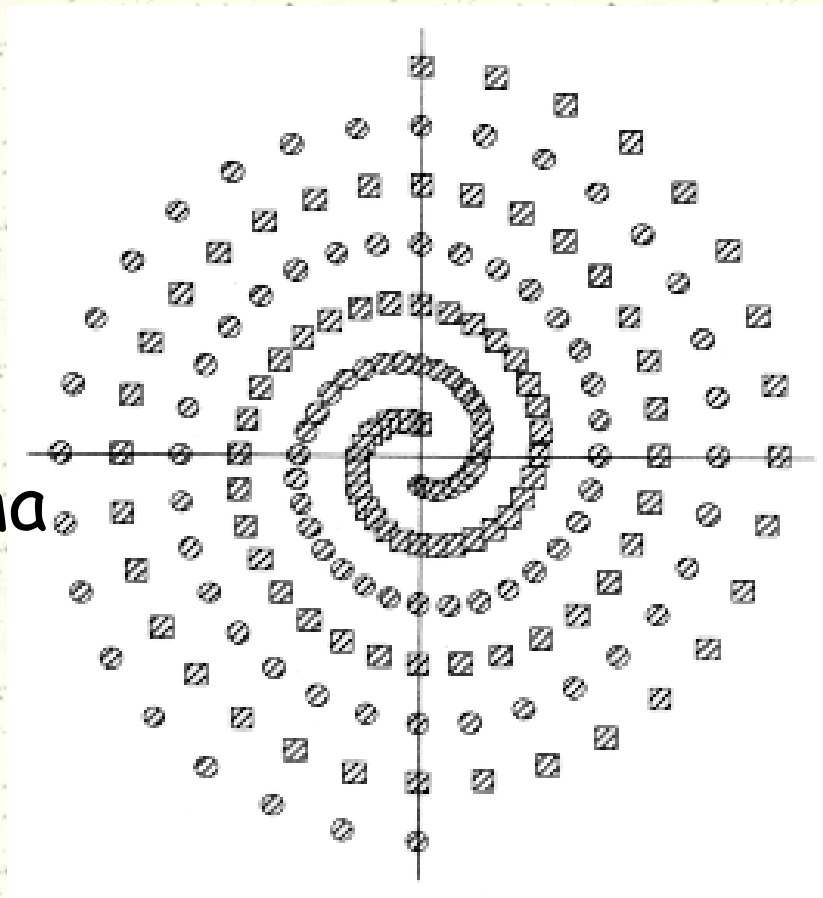
- Proteínas são agrupadas em famílias de acordo com a sua função biológica
- Proteínas de mesma família podem compartilhar estruturas semelhantes ou ter estruturas muito diferentes entre si
- O objetivo é descobrir pequenos trechos de sequências (poucos aminoácidos) que caracterizem inequivocamente uma família.

Tsunoda, D.F., Lopes, H.S. Automatic motif discovery in an enzyme database using a genetic algorithm-based approach. **Soft Computing**, v. 10, n. 4, p. 325-330, 2006.

Classificação de pontos Problema da Espiral entrelaçada

- Trata-se encontrar uma função que determine a qual das duas espirais entrelaçadas pertence um ponto representado no plano x,y

- Originalmente o problema foi proposto com duas espirais de 97 pontos cada, porém pode ser ampliado para mais espirais e mais pontos.



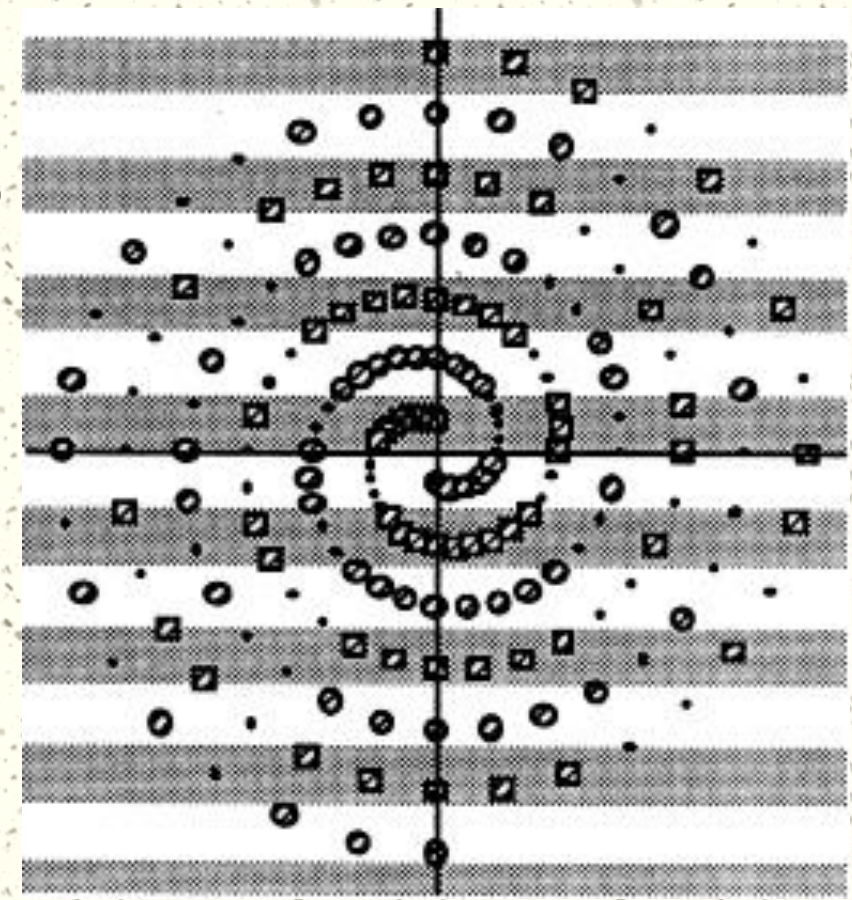
Parâmetros da PG

- T ={x,y, $\Re$}, onde $\Re$ é uma constante aleatória entre -1,000 e +1,000
- F ={+,-,*,%,sen,cos,IFLTE)
  - onde IFLTE significa: IF a ≤ b THEN c ELSE d
  - aridades: 2,2,2,2,1,1,4
- Casos de fitness: 194
- Fitness cru: número de pontos corretamente classificados
- Fitness padronizado: 194-r
- Wrapper: retorna um valor positivo para a classe +1 e mapeia todos os demais valores para a classe -1
- M=10000 com sobreseleção intensa, G=51

Resultados

- Melhor indivíduo da geração 0 acerta 128 pontos
  (SIN (/ Y 0,30400002))

$$f(x,y) = \sin\left(\frac{y}{0{,}30400002}\right)$$

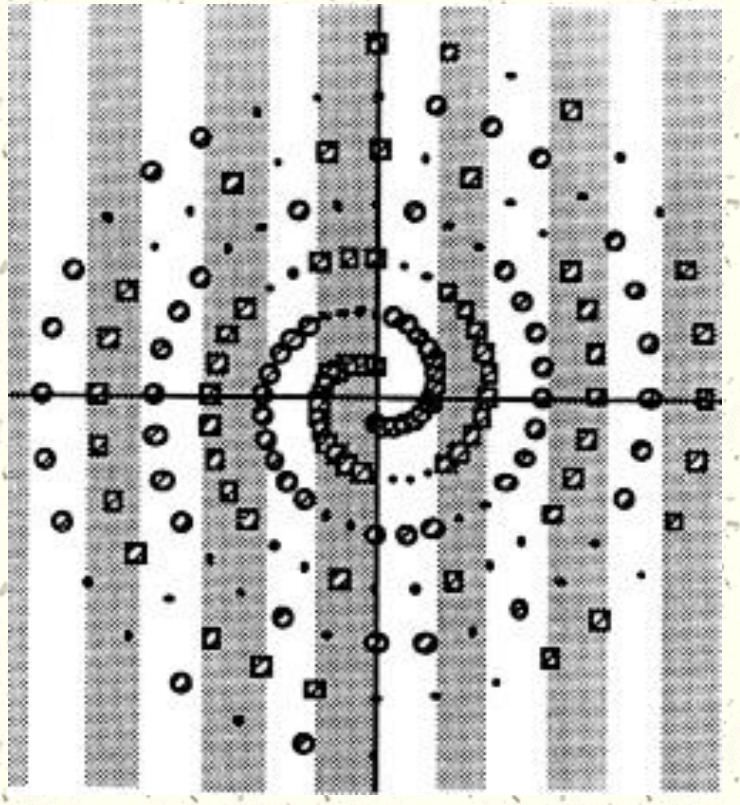


Resultados

- Melhor indivíduo da geração 1 acerta 137 pontos

```
(SEN ( / (SEN X) ( + -
   0,12499994 -0,15999997)))
```

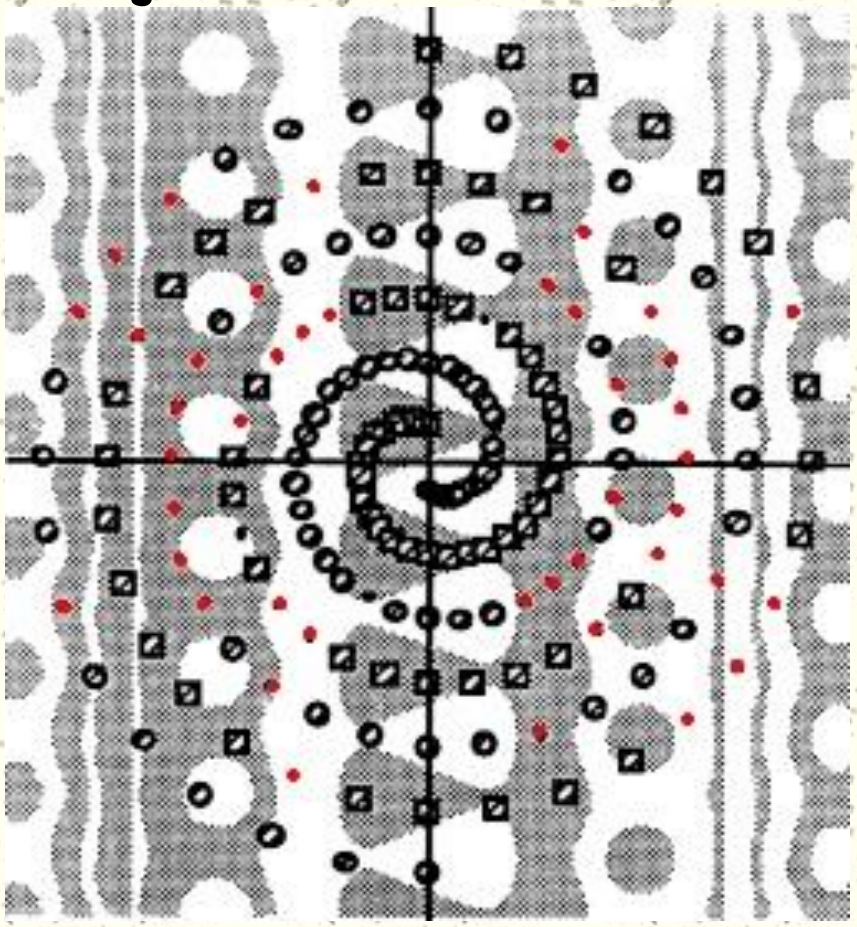
$$f(x,y) = \sin\left(-\frac{\sin(x)}{0{,}28499991}\right)$$



Resultados

- Melhor indivíduo da geração 5 acerta 146 pontos

```
(SEN (- (+ (IF ( * -
0,25699997) ( * X X) (COS
Y) (+ (SEN (COS X)) (+( *
0,18500006 -0,33599997)
(IF Y 0,42000008 X -
0,23399997)))) (SEN (SEN
Y))) (+ (IF ( / (COS X)
(SEN -0,594)) (+ -0,553 Y)
( / Y -0,30499995) (+ Y
X)) (COS ( / X
0,5230001)))))
```



Resultados

- Melhor indivíduo da geração 36 acerta 100% dos pontos

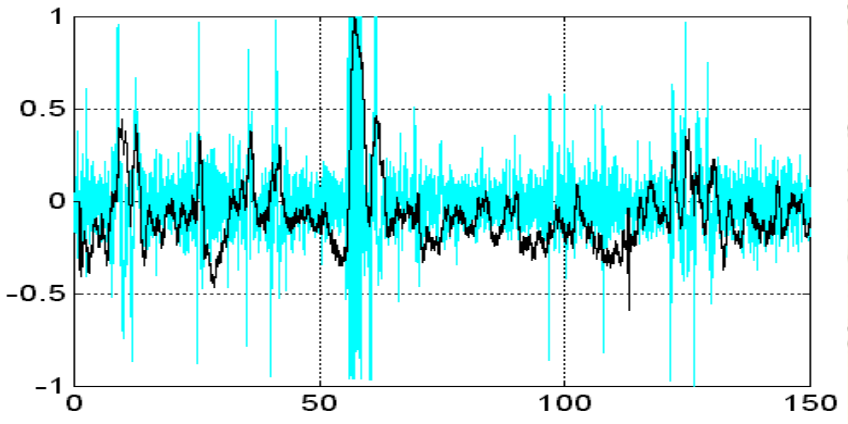
```
(SIN (IFLTE (IFLTE (+ Y Y) (+ X Y) (- X Y) (+ Y Y)) (* X
X) (SIN (IFLTE (% Y Y) (% (SIN (SIN (% Y 0.30400002))) X)
(% Y 0.30400002) (IFLTE (IFLTE (% (SIN (% (% Y (+ X Y))
0.30400002)) (+ X Y)) (% X -0.10399997) (- X Y) (* (+
-0.12499994 -0.15999997) (- X Y))) 0.30400002 (SIN (SIN
(IFLTE (% (SIN (% (% Y 0.30400002) 0.30400002)) (+ X Y))
(% (SIN Y) Y) (SIN (SIN (SIN (% (SIN X) (+ -0.12499994
-0.15999997))))) (% (+ (+ X Y) (+ Y Y)) 0.30400002)))) (+
(+ X Y) (+ Y Y))))) (SIN (IFLTE (IFLTE Y (+ X Y) (- X Y)
(+ Y Y)) (* X X) (SIN (IFLTE (% Y Y) (% (SIN (SIN (% Y
0.30400002))) X) (% Y 0.30400002) (SIN (SIN (IFLTE (IFLTE
(SIN (% (SIN X) (+ -0.12499994 -0.15999997))) (% X
-0.10399997) (- X Y) (+ X Y)) (SIN (% (SIN X) (+
-0.12499994 -0.15999997))) (SIN (SIN (% (SIN X) (+
-0.12499994 -0.15999997)))) (+ (+ X Y) (+ Y Y)))))))) (% Y
0.30400002))))).
```

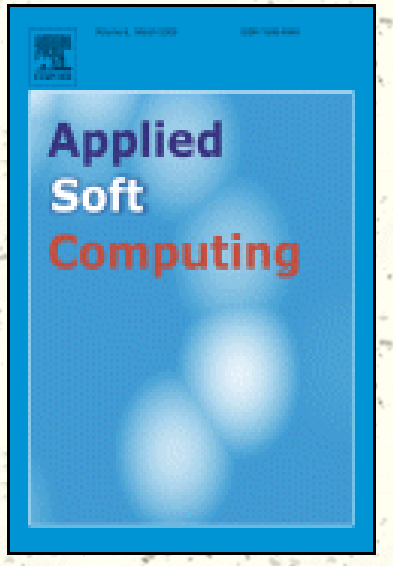
Aplicação prática: detecção de eventos epilépticos

- Epilepsia é uma doença neurológica comum que afeta grande parte da população.
- O diagnóstico é sintomático e baseado principalmente na análise do traçado do Eletroencefalograma (EEG).
- O EEG captura sinais de baixíssima amplitude resultantes da atividade cerebral
- A interpretação do EEG é bastante complexa

Aplicação prática: detecção de eventos epilépticos

- O sinal elétrico do EEG (1 canal) é dividido em janelas de 1 segundo. Para cada uma, um conjunto de parâmetros (features) é calculado
- PG é utilizada para evoluir uma expressão matemática para identificar dois tipos de eventos epilépticos





Lopes, H.S. Genetic programming for epileptic pattern recognition in electroencephalographic signals. **Applied Soft Computing**, v. 7, p. 343-352, 2007