

TUDEL

A grande vantagem da computação evolutiva, é que ao usar um engenho canônico (sem hibridização e sem busca local), com alfabeto binário e operadores convencionais, poucas coisas precisam ser feitas.

- Estabelecer o genótipo e a sua relação com o fenótipo
- Buscar um engenho evolutivo canônico
- Escrever ou obter uma função objetivo adequada
- Certificar o gerador de (pseudo) aleatórios
- Pôr tudo a rodar junto

O problema F6 O tudel foi utilizado na análise da função F6, conforme definido em [Dav91]. Esta função reúne condições boas de testar uma implementação de algoritmo genético, uma vez que ela é oscilante, multimodal, com máximo global único, simétrica e assintótica aos dois lados. Esta função tem como fórmula:

M = 0.5 - (sen√(x² + y²)² - 0.5 / (1.0 + 0.001 × (x² + y²)²)

Seu gráfico pode ser assim visualizado:

Eis algumas características do funcionamento da função F6

- Quebra o número binário de 44 bits em 2 números de 22 bits
- Transforma-os em números decimais
- Multiplica-os por 0.00004768372718899898. Esta multiplicação transforma o range dos números de 0 a 2²² - 1, para o intervalo 0 - 200
- Diminui 100 de cada número. O novo intervalo está entre -100 e 100
- Aplica estes 2 números na fórmula dada.

O objetivo é otimizar M (achar seu ponto máximo) Schaffer et alli. escolheram a função F6 para medir a efetividade dos algoritmos genéticos devido a algumas características interessantes: (1) tem um ponto ótimo, (2) é simétrica, (3) é assintótica à esquerda e à direita. (4) é oscilante (o que leva à breca com tentativas tipo subida da montanha; levam a ótimos locais), (5) tem um só ótimo global.

Entretanto, não há nada de mais especial com F6. Ela pode ser substituída por qualquer outra função.

Ela tem máximo em 1, com valor de x e y igual a 0,5. Logo em ambos os lados, a função decai rapidamente a zero, e depois há uma série de máximos locais cada vez menores, até as assíntotas nos dois lados.

Como o comportamento da função é de antemão conhecido, o objetivo passa a ser conhecer o funcionamento do algoritmo e fazer análises sobre o seu desempenho. Nesse ponto temos uma dificuldade. Olhando por cima, apresentam-se as seguintes possibilidades de teste (quadro obtido por estimativa e usando apenas o senso comum)

Variável	V.inicial	V.final	Intervalo	Número de estados possíveis
Número indivíduos	20	150	130	14
Bits / indivíduo	32	64	32	9
Número gerações	30	60	30	7
Tipos seleção	-	-	-	-
Tipos normalização	0.5	-	-	3
V. decr. p/ janelam.	0.9	0.1	0.8	10
Elitismo	0	10	10	5
Taxa mutação	0.01	0.40	0.39	20
Taxa crossover	0.5	1.0	0.5	10
Tipos crossover	0.1	-	-	2
Pontos crossover	1	8	7	8

Mediante essa análise estimativa, julga-se necessário efetuar 14 × 9 × 7 × 4 × 3 × 10 × 5 × 20 × 10 × 2 × 8 = 1.693.440.000 testes, a fim de poder fazer uma análise de correlação e determinar o que é importante e o que é acessório

na análise do desempenho do algoritmo. Se cada teste levou em média 15 minutos, estima-se que seriam necessários 48.325 anos para concluir essa bateria de testes.

Trata-se de um universo amostral muito grande. Tanto é que já houve pesquisadores (Weinberg, por exemplo) que utilizaram algoritmos genéticos para estabelecer parâmetros adequados para algoritmos genéticos. Aqui optou-se (um tanto imperialmente, é verdade) pela seguinte estratégia

- Primeiro determinar quais normalizadores e seletores são mais adequados, incluindo elitismo, usando parâmetros de tamanho (de indivíduo e de população) e taxas (de mutação e de crossover) dentro do que a literatura sugere.
- Num segundo momento, e tendo estabelecido a melhor configuração de normalizador e seletor mais adequada PARA ESTE PROBLEMA, escolher taxas de mutação e de crossover de melhor desempenho.
- Sabendo as taxas PARA ESTE PROBLEMA, conduzir testes variando os tamanhos, buscando responder a questão de até quanto tais tamanhos podem ser diminuídos, sem que se perca qualidade na resposta.
- Finalmente, de posse de todos os parâmetros corretos – ou pelo menos satisfatórios – para o problema, ele deverá ser executado um bom número de vezes a fim de diminuir o efeito estocástico das respostas, substituindo populações de indivíduos por populações de médias.

Agindo dessa maneira, crê-se poder reduzir o trabalho de sugerir parâmetros adequados a algo mais razoável, embora sempre haja o risco de, por não efetuar uma varredura completa sobre o espaço amostral, ficar preso em algum máximo local.

APL2

A linguagem APL2, na qual foi feito (e será rodado) o TUDEL é uma linguagem bastante antiga, sua primeira versão, o VSAPL é de 1963. Ela nasceu como uma notação matemática mais poderosa e menos ambígua que a matemática convencional. Ela nasceu do trabalho de um professor de matemática canadense de nome Keneth Iverson. Sua proposta original era a de produzir uma nova notação matemática, menos sujeita às ambigüidades da notação convencional.

Na década de 60, trabalhando na IBM em conjunto com Adin Falcoff, ambos produziram a primeira versão de APL, quando um interpretador da linguagem ficou disponível.

A principal característica de APL é o uso de um conjunto especial de caracteres que incluem algumas letras gregas (rho, iota...), símbolos matemáticos convencionais (o sinal de vezes, o de dividido...) e alguns símbolos especialmente inventados.

Este fato sempre limitou a disseminação da linguagem. Até o advento das interfaces gráficas (windows, por exemplo), exigia-se um hardware especial para poder programar em APL.

Programas em APL em geral sempre são muito pequenos, embora poderosos. A linguagem está preparada para tratar arranjos de grandes dimensões. Por exemplo, quando em APL se escreve A+B, se A e B forem escalares (isto é um número único), a resposta também o será. Se A e B são vetores de 100 números, a resposta também o será. Idem para matrizes e até arrays-nd. Em algumas versões de APL este n chega a 256 dimensões.

O APL pode funcionar como uma calculadora. Carregado o programa, qualquer comando fornecido pelo operador será resolvido e o resultado devolvido pelo APL.

Experimente os comandos: 1+1 1 2 3 4 5 +4 1 2 3 4 5 +4 5 6 7 8 1 2 3 4 5 *3 1 2 3 4 5 ? 4 5 6 7 8 1 2 3 4 5 ρ 4 5 6 7 8 ...

Caracteres especiais do APL Para programar em APL é necessário gerar alguns caracteres que não existem no teclado padrão. Eis alguns:

car.	monádico	diádico	como é gerado
ρ	devolve a forma	reformatar dados	ALT-R
⌊	devolve a matriz inversa	resolve o sistema	CTRL-+
⌵	devolve os índices ordenados	segundo um critério	CTRL-3
ι	gera índices	busca em tabelas	ALT-I

Workspace e diferença entre função e variável Tudo o que o usuário define no APL (funções e variáveis) fica armazenado num espaço chamado workspace. Este espaço é salvo (jsave) e carregado (jload) quando se quer interromper o trabalho.

Uma função é um programa escrito pelo usuário que, depois de pronto passa a se comportar como se fosse (e é) mais uma primitiva da linguagem.

Uma variável pode ser numérica ou "caracter", é criada pelo símbolo ← que no teclado é chamado com ALT-acento agudo e para olhar o seu conteúdo basta escrever o nome da variável e dar <enter>.

TUDEL

O TUDEL foi desenvolvido em 1996 em SHARP APL e convertido em 2006 para APL2, modificando-se apenas as interfaces com arquivos. Ele é conversacional e tem os seguintes parâmetros.

TUDEL 2.03 - Pacote de GA - (c) 1996, P.Kantek - UFSC/CPGEEL/LCMI
Para usar este programa, chame a função "tudel" com os seguintes parâmetros
A tudel B C D E F G H I J K L M N O P Q, onde:
... explicações sobre o uso...

fun tudel ind bit ger s n v1 v2 eli txm txc tipc ptcr in mos s.al. nrd
fun00 tudel 100 44 40 1 1 0 0 2 0.01 0.9 1 3 0 4 99999 123



Para você fazer

1. Instale-se no APL2. Aprenda a entrar (Duplo click no ícone) e a sair fazendo)OFF.
2. Carregue o WS TUDEL (faca)LOAD TUDEL).
3. Chame a função "modelo", que descreve a maneira de uso.
4. Descrição dos parâmetros.

A= função objetivo. Neste caso, tem-se

$$\nabla r \leftarrow \text{FUNOBJ } x; x1; x2; x11; x21; x3$$

$$[1] x1 \leftarrow 22 \uparrow x$$

$$[2] x2 \leftarrow 22 \downarrow x$$

$$[3] x11 \leftarrow -100 + 4.76837271889E^{-5} \times (22 \rho 2) \perp x1$$

$$[4] x21 \leftarrow -100 + 4.76837271889E^{-5} \times (22 \rho 2) \perp x2$$

$$[5] x3 \leftarrow (x11 * 2) + x21 * 2$$

$$[6] r \leftarrow (0.5 - (((1 \circ x3 * 0.5) * 2) - 0.5) \div (1 + 1E^{-3} \times x3) * 2), x11, x21$$

$$\nabla$$

B= quantidade de indivíduos

C= número de bits de cada indivíduo

D= número de gerações a processar

E= Tipo de seleção. Pode ser

1. Amostragem estocástica com restituição: a aptidão de todos os indivíduos é usada como parâmetro para uma roda da roleta convencional.
2. Amostragem determinística: Este processo é feito em duas etapas. Na primeira a probabilidade de cada indivíduo ser selecionado é obtido de $p_{select} = \frac{f_i}{\sum f_i}$, ou seja a aptidão daquele indivíduo dividida pela soma de todas as aptidões. O número de cópias deste indivíduo e_i é então obtido a partir de $e_i = n \times p_{select}$, onde n é o tamanho do pool de pais. A parte inteira de e_i gera imediatamente a quantidade de cópias indicada. Na segunda parte do processo, os indivíduos são classificados em ordem decrescente pela parte fracionária de e_i . Os candidatos necessários até preencher o pool são obtidos do topo desta lista ordenada.
3. Amostragem remanescente estocástica com substituição: Também é feita em duas etapas. A primeira é igual ao acima descrito. A segunda, usa uma roda da roleta em que a fatia de cada indivíduo é proporcional à parte fracionária de e_i .
4. Amostragem remanescente estocástica sem substituição: Novamente feito em duas etapas. A primeira segue igual. A segunda usa a parte fracionária de e_i como expectativa (probabilidade de sucesso) em uma operação de caracora. Se houver sucesso, o elemento vai para o pool. Se não houver, passa-se ao próximo.

F= tipo de normalização. Pode ser

- 1 O fitness é igual à função objetivo
- 2 Haverá um janelamento (valores menores que o mínimo, ganham o mínimo)
- 3 Normalização linear

G= Só deve ser diferente de zero se F=3. Neste caso é o valor do melhor indivíduo

H= Se F=1, este aqui deve valer zero. Se F=2 é o mínimo do janelamento. Se F=3, é o decremento a cada novo indivíduo.

I= Número de indivíduos que sofrem elitismo. Varia entre 0 e n, mas obrigatoriamente deve ser par.

J= Taxa de mutação (varia entre 0.000 e 1.000)

K= Taxa de crossover (varia entre 0.000 e 1.000)

L= tipo de crossover. Pode ser 1=n cortes ou 2=uniforme

M= número de pontos (usado apenas se L = 1)

N= modo de inicialização. Se 0=aleatório, se 1=busca em Δ COMECO

O= número de indivíduos (os melhores) a mostrar na listagem

P= semente aleatória (entre 1 e 100000000)

Q= numero da rodada

5. Rode uma vez o modelo conforme proposto para ver o que acontece.
6. Analise da melhor seleção: para E=1, rode 3 vezes mudando a semente aleatória, e verifique o fitness médio

1	2	3

para E=2, rode 3 vezes mudando a semente aleatória, e verifique o fitness médio

1	2	3

para E=3, rode 3 vezes mudando a semente aleatória, e verifique o fitness médio

1	2	3

para E=4, rode 3 vezes mudando a semente aleatória, e verifique o fitness médio

1	2	3

Pelos resultados vistos, a melhor seleção é a _____.

7. Se quiser, especule e descubra o funcionamento do conceito de normalização e janelamento. Neste caso, faça experiências variando o uso (ou não) de normalização, o conceito de valor mínimo e o de janelamento.

Caso contrário, faça F=1 (função objetivo=fitness), G=0 e H=0 e siga o exercício.

8. Escolha de crossover Determine, E, F, G e H e agora varie L=2

1	2	3

Determine, E, F, G e H e agora varie L=1 e M=2

1	2	3

Determine, E, F, G e H e agora varie L=1 e M=5

1	2	3

Conclusão: o melhor crossover é _____.

9. Elitismo (os melhores desta geração estarão garantidos na próxima). Faça I=0

1	2	3

Faça I=2

1	2	3

Faça I=5

1	2	3

Conclusão: O melhor elitismo é _____.

10. Tamanho de população Faça B=200

1	2	3

Faça B=50

1	2	3

Conclusão: O melhor tamanho é _____.

11. Taxas Varie a taxa de mutação, faça J=0.01

1	2	3

Mutação, faça J=0.03

1	2	3

Mutação, faça J=0.06

1	2	3

12. Se quiser, faça outros experimentos. Pense a respeito do que poderia ser testado e variado. Há diversão para meses aqui...

13. Para concluir, quais os melhores parâmetros?
