Um algoritmo evolucionário híbrido baseado na fertilização in vitro para regressão simbólica

Ewerton Carlos de Araujo Assis[1][0000−0003−1767−1735]

Universidade Federal do ABC, São André, SP, Brasil carlos.assis@ufabc.edu.br

Abstract. A partir de soluções meta-heurísticas da computação evolucionária e soluções da programação genética, o presente projeto de pesquisa tem por finalidade construir uma solução algorítmica com a finalidade de resolver problemas comumente solucionados com o uso da regressão simbólica. Várias propostas de soluções já foram apresentadas na literatura, usando abordagens diferentes. Diferentes *benchmarks* estão disponíveis com a finalidade de comparar essas diferentes soluções, em termos de performance e efetividade das soluções propostas. A proposta apresentada constrói um algoritmo evolucionário híbrido, usando abordagens usuais da computação evolucionária e da programação genética, a fim de solucionar problemas no escopo da regressão simbólica. É utilizada uma abordagem baseada em fertilização in vitro (FIV), na qual, a cada iteração do algoritmo, uma nova solução é construída tendo por base as duas melhores soluções disponíveis na população, que em seguida será inserida na população final. A solução algorítmica será avaliada com base no *benchmark* SRSD (Symbolic Regression Datasets and Benchmarks for Scientific Discovery) e comparada com PySR (*High-Performance Symbolic Regression in Python and Julia*), *gplearn*; e *Transformation-Interaction-Rational*.

Keywords: Regressão Simbólica · Meta-heurística · Programação Genética.

1 Introdução

A análise de regressão (*regression analysis*) é uma técnica estatística utilizada nas áreas de Aprendizado de Máquina e Inteligência Artificial que tem por finalidade estabelecer uma relação entre uma variável dependente (ou alvo) e uma ou mais variáveis independentes. Neste contexto, a regressão simbólica (*symbolic regression*) é uma técnica de análise de regressão que explora o espaço de expressões matemáticas a fim de encontrar um modelo (ou expressão) que melhor se ajusta a um conjunto de dados, sem ter um modelo previamente construído que estabeleça a relação entre as variáveis deste conjunto de dados. As expressões são construídas iterativamente, combinando operadores matemáticos, funções analíticas, constantes e variáveis, que são então avaliadas como expressões candidatas, tendo como critério uma função de aptidão (ou *fitness*).

Esta técnica é comumente utilizada em áreas como ciências naturais e engenharias, nas quais frequentemente não existe um modelo que estabeleça relações

entre as variáveis de um conjunto de dados e, a partir da análise de regressão e da regressão simbólica, pode-se extrair leis empíricas e modelos (expressões) a partir de dados experimentais.

Dado que as expressões são construídas a partir de uma busca em um espaço de busca de expressões, tendo como guia nessa busca uma função de aptidão que determinará a obtenção de uma expressão que melhor se ajuste ao modelo de relação entre as variáveis do conjunto de dados, técnicas diversas são utilizadas para resolver este tipo de problema, sendo uma delas a Programação Genética (PG; *Genetic Programming*).

A PG é uma técnica da Computação Evolucionária [6], baseada em uma população de soluções candidatas que são avaliadas conforme sua aptidão a melhor representar uma expressão matemática que represente o modelo de relação entre as variáveis do conjunto de dados. Cada indivíduo da população, portanto, representa uma expressão matemática (ou um código de programação), geralmente codificado como uma árvore sintática. Iterativamente, novas populações de indivíduos-soluções são construídas, com base na população anterior, que é novamente avaliada conforme uma função de aptidão, que poderá penalizar soluções que sejam excessivamente complexas (apresentando, portanto, *overfitting*).

Soluções da Computação Evolucionária, que comumente se apresentam como soluções algorítmicas baseadas em populações de soluções candidatas, podem apresentar deficiências frente a outras técnicas, como o tempo de convergência e os diversos efeitos que a aleatoriedade desses modelos pode apresentar sobre a solução algorítmica [6][9]. A fim de otimizar e apresentar abordagens novas em busca de soluções melhores, diversas técnicas da Computação Evolucionária foram apresentadas em conjunto com outras heurísticas a fim de solucionar problemas de otimização a partir de um espaço de busca. As meta-heurísticas, portanto, apresentam mecanismos sofisticados junto a heurísticas já bem estabelecidas na literatura acadêmica [15].

O presente trabalho tem por finalidade utilizar um mecanismo de variabilidade baseado em fertilização in-vitro (FIV) [2][3] no contexto da PG e da Regressão Simbólica. A fim de comparar os ganhos e perdas dessa técnica híbrida entre PG e FIV, outras soluções encontradas na literatura são utilizadas como *benchmark*: a PG clássica a partir da biblioteca *gplearn*[13], apresentada por Matsubara et. al. [12]; Algoritmo Evolucionário (AE; *Evolutionary Algorithm*) através da biblioteca *PySR* [5][4], também apresentada por Matsubara et. al. [12]; e *Transformation-Interaction-Rational* [8]. Estas soluções são comparadas a partir de um *benchmark* reconhecido na literatura: 120 conjuntos de dados criados a partir das *Feynman Lectures on Physics* [12].

Este trabalho é organizado da seguinte forma: a seção 2 apresenta uma definição formal de problemas em regressão simbólica e uma breve introdução à programação genética e ao uso de soluções híbridas e meta-heurísticas; a seção 3 apresenta a metodologia na construção da solução proposta e no uso do *benchmark* escolhido; a seção 4 analisa os resultados das avaliações feitas; e, por fim, a seção 5 conclui o trabalho, apresentando desenvolvimentos que podem ser feitos como consequência dessa pesquisa.

2 Regressão Simbólica, Meta-heurísticas e Programação genética

A Regressão Simbólica (RS) [16] é uma técnica de modelagem matemática orientada por dados cujo objetivo é descobrir, simultaneamente, a estrutura funcional e os parâmetros de uma expressão matemática que melhor represente a relação entre variáveis observadas em um conjunto de dados. A regressão simbólica busca no espaço de todas as expressões matemáticas válidas uma função $f : \mathbb{R}^d \to \mathbb{R}$ tal que $f(\mathbf{x}) \approx y$, onde $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ são os preditores e $y \in \mathbb{R}$ é a variável resposta.

Formalmente, define-se um *espaço de busca simbólico* $\mathcal{F}$, construído a partir de um conjunto de primitivas $\mathcal{P} = \mathcal{F}_b \cup \mathcal{T}$, onde:

- $\mathcal{F}_b$ é um conjunto finito de operadores ou funções (por exemplo, $+$, $-$, $\times$, $\div$, sin, exp);
- $\mathcal{T}$ é o conjunto de terminais, que inclui variáveis do domínio $x_1, \ldots, x_d$ e constantes numéricas.

Cada função candidata $f \in \mathcal{F}$ pode ser representada por uma *árvore sintática* (ou *parse tree*), onde os nós internos correspondem a funções em $\mathcal{F}_b$ e as folhas pertencem a $\mathcal{T}$. O espaço $\mathcal{F}$ é composto por todas as árvores sintaticamente válidas geradas por combinações finitas desses elementos, possivelmente com profundidade ou aridade máxima restrita para controle de complexidade.

O problema da regressão simbólica consiste, então, em resolver:

$$\min_{f \in \mathcal{F}} \ \mathcal{L}(f) + \lambda \cdot \Omega(f),$$

onde:

- $\mathcal{L}(f)$ é uma função de perda (por exemplo, o erro quadrático médio sobre um conjunto de dados $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$);
- $\Omega(f)$ é uma medida de complexidade do modelo (como o tamanho ou profundidade da árvore);
- $\lambda \geq 0$ é um hiper-parâmetro que controla o equilíbrio entre ajuste e simplicidade (conhecido como *bias-variance trade-off*).

A regressão simbólica se destaca por produzir modelos explicitamente interpretáveis, ao contrário de métodos de aprendizado de máquina de natureza opaca ou não explicável [10]. Esta característica é particularmente relevante em contextos científicos e de engenharia, nos quais a transparência do modelo é tão ou mais importante que sua acurácia preditiva.

Este processo de busca é normalmente realizado por algoritmos estocásticos ou heurísticos. Uma solução heurística executa uma busca iterativa guiada por uma função de aptidão derivada de $\mathcal{L}(f) + \lambda \cdot \Omega(f)$, explorando o espaço $\mathcal{F}$ por meio de operadores definidos por uma determinada heurística ou meta-heurística.

Soluções heurísticas são geralmente construídas para resolver classes de problemas específicos, tendo em conta as características e desafios desta classe de

problemas. No caso da Regressão Simbólica, vimos que o espaço de busca de uma solução-modelo que represente uma expressão matemática entre as variáveis do conjunto de dados pode se tornar complexo, tendo em vista o tamanho e a complexidade destes conjuntos de dados. Encontrar uma expressão que estabeleça essas relações, portanto, passa necessariamente pela construção de uma solução heurística, com vista à classe do problema analisado.

Michel Gendreau apresenta meta-heurísticas como "métodos de solução que orquestram uma interação entre procedimentos de melhora local e estratégias de alto nível com o fim de criar um processo capaz de escapar de ótimos locais e realizar uma busca robusta em um espaço de busca" (tradução livre) [9]. Meta-heurísticas "provêm resultados aceitáveis, em um tempo razoável, como solução de problemas difíceis e complexos, nas Ciências e na Engenharia" (tradução livre) [15]. Portanto, são soluções algorítmicas que podem ser reutilizadas e reconstruídas a partir de modificações mínimas.

Algorithm 1: Algoritmo canônico da Programação Genética

Data: População P de tamanho μ aleatoriamente construída e *máximo_gerações* representando o número de iterações na construção de novas populações

Result: População P de tamanho μ de descendentes

```
1 contador_gerações ← 0;
2 repeat
3 |  Calcula a aptidão f(p) para cada indivíduo p ∈ P;
4 |  Seleção de indivíduos pais p ∈ P para recombinação;
5 |  Aplica uma técnica de recombinação entre os pais, gerando novos
  |   indivíduos em P′;
6 |  Realiza a mutação de indivíduos p ∈ P′, respeitando uma técnica
  |   específica de mutação;
7 |  Seleciona μ indivíduos dentre P e P′ que irão compor a nova geração
  |   P^n;
8 |  contador_gerações ← contador_gerações + 1;
9 until contador_gerações < máximo_gerações;
```

Conforme anteriormente exposto na Introdução, a Programação Genética é uma técnica da Computação Evolucionária [6], uma heurística baseada em uma população de indivíduos-soluções para um determinado problema, tendo em vista determinado espaço de busca S. Diferentemente de outros algoritmos canônicos da Computação Evolucionária que representam seus indivíduos como vetores de genes-variáveis, a PG tem como representação dos indivíduos códigos ou programas de computadores, comumente representados por uma árvore sintática. Um conjunto populacional P é inicialmente gerado através de uma abordagem específica (que pode ser aleatória, utilizando construções válidas de uma árvore sintática que represente uma expressão matemática, por exemplo), e a cada geração ou iteração do algoritmo, uma nova população P' é criada, a qual substitui a população parental P. A população parental P é substituída pela população de

herdeiros P' seguindo a função de aptidão (*fitness*) $f : P \to \mathbb{R}$, que determina a qualidade dos indivíduos de uma população. A cada novo indivíduo gerado $p \in P'$, a função de aptidão é aplicada para encontrar a qualidade ou *suitability* daquele indivíduo $f(p)$.

Algorithm 2: Solução híbrida (*IVF/PG-SR*)

Data: População P de tamanho μ aleatoriamente construída e *máximo_gerações* representando o número de iterações na construção de novas populações
Result: População P de tamanho μ de descendentes

```
1  contador_gerações ← 0;
2  repeat
3      superpais ← melhores ρ' = μ * 25% indivíduos em P da geração
         corrente;
4      embrião ← manipulação genética dos superpais;
5      Seleciona ρ = μ/2 pais em P para reprodução;
6      Construção de um conjunto vazio de descendentes P';
7      for i → 0 to ρ do
8          descendente ← recombinação do pai 2 * i com o pai 2 * i + 1;
9          Mutação realizada sobre descendente (busca local);
10         Adição de descendente ao conjunto P' de descendentes;
11     end
12     O conjunto P' de descendentes gerados substitui a geração parental
         e passa a ser a população corrente;
13     Elitismo: o pior indivíduo dentre os descendentes gerados é
         substituído pelo melhor indivíduo da geração parental;
14     Transferência do embrião para a população de novos indivíduos P';
15     contador_gerações ← contador_gerações + 1;
16 until contador_gerações < máximo_gerações;
```

Os componentes lógicos do algoritmo canônico da Programação Genética são: (1) seleção para reprodução, ou a escolha de indivíduos-pais para recombinação; (2) cruzamento ou recombinação (*crossover*), que realiza a combinação de dois indivíduos previamente selecionados: as subárvores são trocadas, a partir de estratégias próprias de corte dessas subárvores; (3) mutação, que altera a subárvore de um indivíduo-solução por uma subárvore aleatoriamente gerada, respeitando-se parâmetros próprios para essas alterações; e (4) técnicas de seleção dos indivíduos que irão compor a nova população após uma iteração — por exemplo, elitismo, que seleciona os melhores indivíduos em P e P' para compor a população final da iteração n, P^n, respeitando-se $|P| = |P'| = |P^n|$.

Geralmente, o critério de parada das iterações é um número específico de iterações (chamado de número máximo de gerações), mas outros critérios podem ser estabelecidos, como atingir um platô, no qual nenhum indivíduo novo gerado apresenta qualidade melhor a outros indivíduos gerados, após um determinado

número de gerações; ou os indivíduos-soluções atingem um determinado nível de aptidão previamente definido.

O algoritmo 1, na página 4, apresenta uma versão do algoritmo canônico da Programação Genética.

Soluções meta-heurísticas emergem, portanto, com o uso de diferentes heurísticas para solucionar classes de problemas complexos e variados. Uma dessas meta-heurísticas apresentadas na literatura baseia-se em um mecanismo de seleção inspirado na fertilização in-vitro, que realiza a manipulação de soluções a fim de gerar indivíduos-soluções a partir dos melhores indivíduos de uma população de soluções. O algoritmo 2, na página 5, apresenta uma solução heurística construída a partir da PG e da heurística inspirada na FIV (FIV/PG-RS) para solucionar problemas em Regressão Simbólica.

3 Metodologia

A fim de realizar uma avaliação da heurística construída em FIV/PG-RS, as soluções apresentadas pelo trabalho *"Symbolic Regression Datasets and Benchmarks for Scientific Discovery"* [12] foram utilizadas como base de comparação; notavelmente, o *gplearn* [11] e o *PySR* [4]. Estas duas soluções foram escolhidas pelo fato de o *gplearn* apresentar o pior desempenho na avaliação com outras soluções, e o *PySR* apresentar o melhor desempenho [12]. Além dessas duas soluções algorítmicas, o *Transformation-Interaction-Rational* (TIR) [8] também é utilizado como solução base de avaliação, por ser uma solução recente e que não foi contemplada no estudo original.

O *dataset* utilizado como meio de comparação é composto por 120 conjuntos de dados criados a partir das *Feynman Lectures on Physics*, organizados em três subconjuntos de dados, classificados conforme a complexidade destes: complexidade baixa (*easy*), complexidade média (*medium*) e complexidade alta (*hard*), consistindo, respectivamente, de 30, 40 e 50 problemas distintos. A complexidade é definida a partir do número de operações necessárias para obter a equação original e do intervalo dos domínios de amostragem.

As soluções apresentadas para o *gplearn* e o *PySR* estão disponibilizadas no estudo mencionado [12]. A solução *Transformation-Interaction-Rational* foi utilizada conforme código disponibilizado pelo autor, com ajustes para as análises apresentadas por Matsubara et. al. [12]. A solução heurística FIV/PG-RS foi criada a partir da solução apresentada para o *gplearn*, seguindo os parâmetros de configuração originais, com alterações apresentadas no algoritmo 2, na página 5.

Temos, portanto, quatro soluções a serem executadas sobre o *dataset* composto por 120 conjuntos de dados. As execuções foram realizadas em instâncias EC2 (*Elastic Compute Cloud*) da Amazon Web Services (AWS), com a seguinte configuração: 16 vCPU (virtual CPU; ou 16 núcleos de processamento), 32 GB de RAM e 20 GB de armazenamento baseado em EBC (Elastic Block Store; SSD). Para cada solução, uma instância isolada foi utilizada para a execução desta.

4 Resultados

Os resultados das soluções foram avaliados utilizando o MSE (*Mean Squared Error*). Para cada conjunto de dados, o MSE foi calculado seguindo a fórmula:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

onde:

- y_i valor real (observado);
- $\hat{y}_i$ valor predito (estimado);
- n número de amostras.

Em seguida, foi feita a média e a mediana dos MSE de todos os conjuntos de dados, para cada solução avaliada. Portanto, temos os seguintes valores para cada solução avaliada:

- *gplearn*: média 9.0970 e mediana 1.0000;
- IV/PG-RS (*ivf-gplearn*): média 9.1293 e mediana 1.0000;
- TIR: média 3.8676 e mediana 0.0000; e
- *PySR*: média 0.5728 e mediana 0.0000.

A seguir, temos a distribuição dos resultados para cada uma das soluções, também utilizando os valores MSE, a partir de gráficos de histograma e diagrama de caixa.

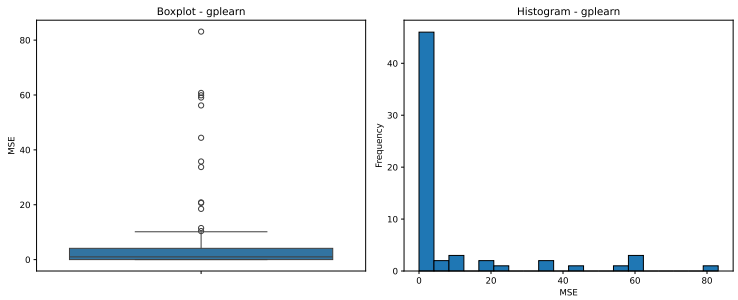

Fig. 1. Distribuição dos valores MSE para o *gplearn*

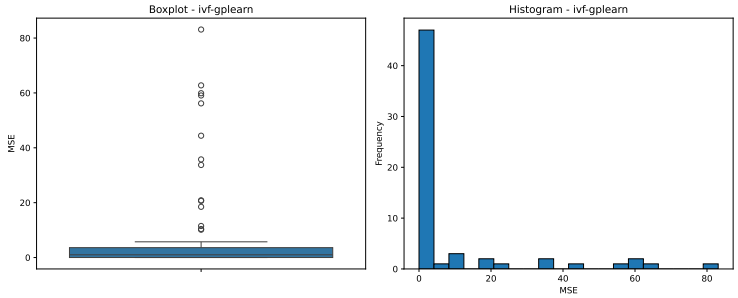

Fig. 2. Distribuição dos valores MSE para o IV/PG-RS (*ivf-gplearn*)

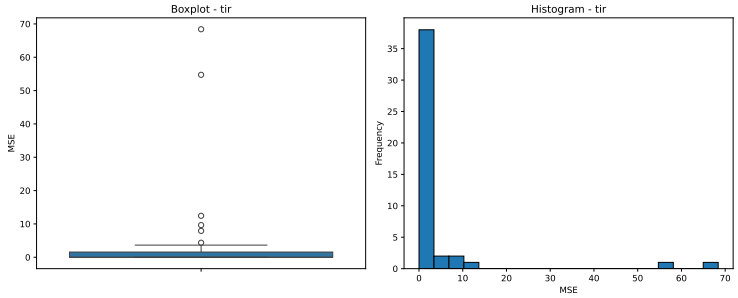

Fig. 3. Distribuição dos valores MSE para o TIR

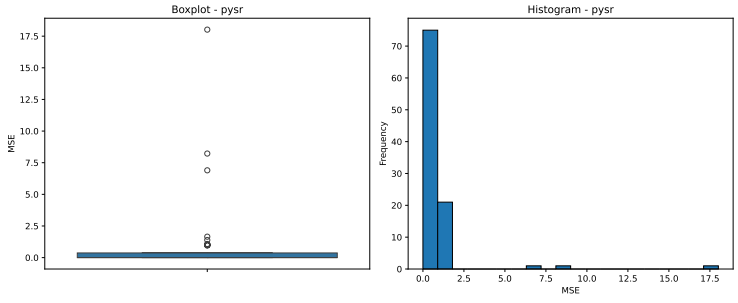

Fig. 4. Distribuição dos valores MSE para o *PySR*

5 Conclusão

A partir dos resultados, foi possível aferir o posicionamento das soluções TIR e FIV/PG-RS frente às soluções *gplearn* e *PySR*, usando o *benchmark* proposto. Ainda que o FIV/PG-RS tenha um resultado próximo ao *gplearn*, este não se mostrou uma boa solução, comparativamente ao TIR e ao *PySR*.

Como trabalho futuro, poderiam ser investigados outros métodos de manipulação genética dos *superpais* (algoritmo 2, na página 5), a fim de gerar variabilidade que seja significativa e que explore melhor o espaço de busca. Além disso, utilizar ferramentas como o Optuna [1] para otimizar os parâmetros de cada solução.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
2. Camilo-Junior, C.G.: Algoritmo Auxiliar Paralelo inspirado na Fertilização in Vitro para melhorar o desempenho de Algoritmos Genéticos. Ph.D. thesis, Universidade Federal de Uberlândia (Março 2010)
3. Camilo-Junior, C.G., Yamanaka, K.: In vitro fertilization genetic algorithm. In: Evolutionary Algorithms, chap. 4, pp. 57–68. InTech (2011)
4. Cranmer, M.: Interpretable machine learning for science with pysr and symbolicregression.jl (2023), https://arxiv.org/abs/2305.01582
5. Cranmer, M.: Pysr: High-performance symbolic regression in python and julia (2025), https://ai.damtp.cam.ac.uk/pysr/
6. De Jong, K.A.: Evolutionary computation: a unified approach. MIT Press, Cambridge (2006)
7. de Franca, F.O., Aldeia, G.S.I.: Interaction–transformation evolutionary algorithm for symbolic regression. Evolutionary Computation **29**(3), 367–390 (09 2021). https://doi.org/10.1162/evco_a_00285, https://doi.org/10.1162/evco_a_00285
8. de França, F.O.: Transformation-interaction-rational representation for symbolic regression. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 920–928. GECCO '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3512290.3528695, https://doi.org/10.1145/3512290.3528695
9. Gendreau, M., Potvin, J.Y. (eds.): Handbook of Metaheuristics. Springer, New York (2010)
10. Harrison, J., Bosman, P.A.N., Alderliesten, T.: Thinking outside the template with modular gp-gomea (2025), https://arxiv.org/abs/2505.01262
11. Koza, J., Poli, R.: Genetic Programming, pp. 127–164. Springer US (01 2005). https://doi.org/10.1007/0-387-28356-0_5
12. Matsubara, Y., Chiba, N., Igarashi, R., Ushiku, Y.: Rethinking symbolic regression datasets and benchmarks for scientific discovery. Journal of Data-centric Machine Learning Research (2024), https://openreview.net/forum?id=qrUdrXsiXX
13. Stephens, T.: gplearn: Genetic programming in python, with a scikit-learn inspired api (2025), https://gplearn.readthedocs.io/

14. Sun, H., Moscato, P.: A memetic algorithm for symbolic regression. In: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 2167–2174 (2019). https://doi.org/10.1109/CEC.2019.8789889
15. Talbi, E.G.: Metaheuristics: from design to implementation. Wiley, Hoboken, New Jersey (2009)
16. Vladislavleva, E.J., Smits, G.F., den Hertog, D.: Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Transactions on Evolutionary Computation **13**(2), 333–349 (2009). https://doi.org/10.1109/TEVC.2008.926486
17. Whitley, D.: An overview of evolutionary algorithms: Practical issues and common pitfalls. Information and Software Technology **43**, 817–831 (2001)
18. Whitley, D.: Genetic algorithms and evolutionary computing. Tech. rep., Computer Science Department, Colorado State University (2002)