GEMS: Genetically Evolving Models in Science

Angelo Pirrone e Fernand Gobet
Centre for Philosophy of Natural and Social Science
London School of Economics and Political Science
Regno Unito

Sommario

Computational scientific discovery is an important area of research in cognitive science. Not only can a scientific theory be represented by a computer program, but it can also be discovered by computers, using techniques and methodologies from artificial intelligence. Here, we present a new methodology currently under development, called GEMS, that has been successfully applied in cognitive science in order to semi-automatically generate scientific theories. GEMS is an application of genetic programming that, given the protocol of an experiment, a set of experimental data and elementary operations (in our case, elementary psychological processes), evolves programs that 'behave' like an experimental subject. From one generation to the next, the programs are improved thanks to evolutionary plausible mechanisms that aim to minimise the discrepancy between model predictions and data. Interestingly, GEMS makes it possible to perform an efficient search in the combinatorial model space; the output of GEMS is a formal scientific theory of a specific dataset, expressed as a computer program. In this paper, we present the main features of GEMS with an example of how it could be applied in a hypothetical scenario; we discuss the theoretical implications of this approach for scientific discovery; and we present ideas for future research in cognitive science using GEMS.

Keywords: genetic programming, scientific discovery, cognitive science

Corrispondenza: {a.pirrone,f.gobet}@lse.ac.uk. Ricerca finanziata dal Consiglio europeo della ricerca (ERC-ADG-835002—GEMS). Il codice di programmazione utilizzato in GEMS è disponibile contattando gli autori.

1. Introduzione

Lo scopo di questo contributo è quello di presentare brevemente un'applicazione dell'Intelligenza Artificiale che consente di scoprire, in maniera semi-automatica, teorie scientifiche nell'ambito delle scienze cognitive. Il nostro lavoro evidenzia le funzionalità pratiche, le modalità d'uso, il contesto di applicazione e le implicazioni teoriche di tale approccio per la scoperta scientifica. È importante sottolineare che questo approccio è nel pieno del suo sviluppo, e quindi rimangono ancora aspetti da perfezionare ed importanti questioni da risolvere, come verrà descritto in conclusione di questo articolo.

La scoperta di una teoria scientifica può essere descritta come una ricerca euristica all'interno dello spazio combinatorio dei modelli che possono aver generato dei dati oggetto di interesse (Frias-Martinez e Gobet, 2007; Pirrone e Gobet, 2020; Simon, 1977). A loro volta, i modelli che possono aver generato dei dati vengono qui descritti come le combinazioni possibili di operatori di base, ovvero operazioni che vengono arbitrariamente considerate irriducibili. I ricercatori spesso comparano una serie di dati raccolti alle predizioni qualitative e/o quantitative di un modello. Più raramente, i dati vengono comparati ad una serie di modelli. Ciò nonostante, un'investigazione esaustiva del completo spazio dei modelli che può aver generato dei dati non viene presa in considerazione, dal momento che lo spazio dei modelli è comunemente molto ampio (se non virtualmente infinito) e questo ha ovvie conseguenze da un punto di vista pratico e computazionale. Si noti che qui non stiamo parlando esclusivamente dello spazio parametrico che caratterizza un modello, ma della combinazione dello spazio dei modelli. Fortunatamente, in Intelligenza Artificiale sono state proposte tecniche e metodologie che possono essere applicate anche al problema in questione. Una di queste, è la Programmazione Genetica (Koza, 1992).

Con questo modesto contributo vogliamo rendere questa metodologia il più possibile accessibile agli scienziati cognitivi, nella speranza che il nostro lavoro possa motivare ricerca ed applicazioni future.

2. Modalità d'uso e contesto d'applicazione

La Programmazione Genetica è una metodologia di programmazione che, partendo da una popolazione di individui (i.e., programmi) iniziali, spesso generati in maniera casuale, evolve gli individui utilizzando principi propri della teoria dell'evoluzione in modo da ottimizzare una funzione di fitness. Nella Programmazione Genetica, ogni individuo viene generalmente espresso utilizzando una struttura ad albero composta da terminali (i.e., gli input) ed operatori (i.e., le operazioni che possono essere effettuate sui terminali), come ad esempio mostrato in Figura 1 nella quale sono rappresentate quattro diverse strutture ad albero. In Figura 1, abbiamo scelto di riportare all'interno di una circonferenza i terminali (A e B) e all'interno di un rettangolo gli operatori. Una spiegazione degli operatori e dei terminali verrà data

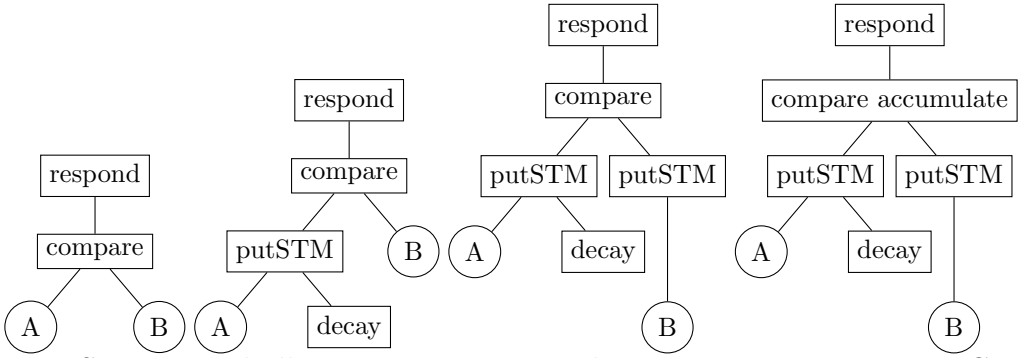

Figura 1. Struttura ad albero comunemente utilizzata in Programmazione Genetica per rappresentare gli individui. Si noti come ogi albero rappresentato in questo caso, può essere sia generato casualmente come individuo della prima generazione o può essere frutto di mutazione o crossover tra due alberi in generazioni successive. Ad esempio, il quarto albero può essere considerato una mutazione del terzo albero, nel quale l'operatore 'compare', tramite mutazione, diventa l'operatore 'compare accumulate'. Lisp, il linguaggio di programmazione utilizzato da Koza (1992) e in generale tra i più utilizzati in Programmazione Genetica e in GEMS, ha come tipo di dato primitivo la struttura ad albero.

in seguito; per il momento entrambi possono essere intesi come elementi astratti ed ipotetici.

I principali meccanismi evolutivi (Koza, 1992; Poli, Langdon, McPhee e Koza, 2008) sono la riproduzione, la mutazione e il crossover. La riproduzione copia un individuo da una generazione all'altra. La mutazione genera un cambiamento casuale in uno o più punti di un albero – ad esempio sostituendo un operatore con un altro operatore disponibile, mentre il crossover seleziona due componenti di due alberi diversi (a seconda di algoritmi diversi, il crossover può avere caratteristiche diverse) e li scambia tra i due alberi.

Questa tecnica ha avuto molte applicazioni in fisica ed ingegneria (Koza, Keane e Streeter, 2004; Lohn, Hornby e Linden, 2004) e sporadiche applicazioni – tutte riconducibili al nostro network di ricerca – nelle scienze cognitive per la generazione di teorie (i.e., modelli) (Addis, Gobet, Lane e Sozou, 2019; Addis, Sozou, Lane e Gobet, 2016; Frias-Martinez e Gobet, 2007; Gobet e Parker, 2005; Lane e Gobet, 2008, 2012, 2013; Lane, Sozou, Addis e Gobet, 2014; Lane, Sozou, Gobet e Addis, 2016; Sozou, Lane, Addis e Gobet, 2017).

Il sistema che noi utilizziamo per la generazione di teorie, viene chiamato GEMS (Genetically Evolving Models in Science) ed è una applicazione della Programmazione Genetica, con alcuni accorgimenti specifici per la scoperta di una teoria scientifica nell'ambito delle scienze cognitive, Figura 2. L'idea alla base è molto semplice: evolvere programmi che svolgano un protocollo sperimentale simile a quello somministrato ai partecipanti umani e producano risultati simili a quelli prodotti dai partecipanti.

In questo caso prenderemo come esempio un semplice ed ipotetico esperimento

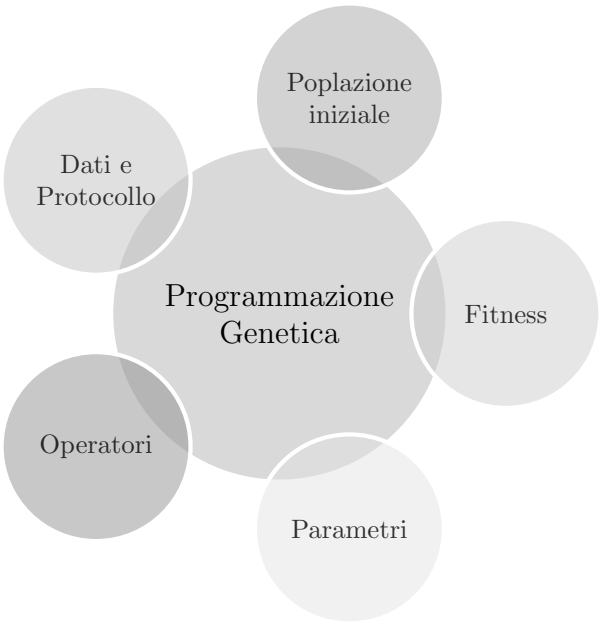

Figura 2. Rappresentazione grafica del sistema GEMS. Meccanismo fondamentale del sistema GEMS è la Programmazione Genetica che, dato un numero di parametri ed operatori, genera una popolazione iniziale e cerca di minimizzare una funzione di fitness sulla base di dati sperimentali e di un protocollo sperimentale. Il risultato della Programmazione Genetica è una teoria cognitiva che definisce in maniera chiara ed inequivocabile il rapporto tra terminali ed operatori nell'aver generato dei dati, dato uno specifico protocollo sperimentale.

simile ad un delayed-match-to-sample (e.g., Chao, Haxby e Martin, 1999) nel quale ai soggetti, in una prima fase, viene presentato uno stimolo A, e dopo una successiva fase di delay, viene presentato uno stimolo B. Compito del partecipante è decidere se lo stimolo B è 'uguale' o 'diverso' rispetto allo stimolo A, premendo uno di due tasti su una tastiera. L'output di questo esperimento sono i tempi di reazione e l'accuratezza della decisione.

Nel voler procedere nel definire il sistema che permette di generare in maniera semi-automatica teorie, il primo passo è quello di definire un *ambiente del compito* (Frias-Martinez e Gobet, 2007) che definisce i dettagli dell'esperimento (e.g., numero di soggetti e protocollo sperimentale). Lo spazio dei possibili modelli (programmi) è definito dalle combinazioni possibili degli *operatori*, ovvero le operazioni cognitive di base, e dei *terminali*, gli input che vengono letti dal sistema. Operatori e terminali interagiscono con l'ambiente del compito per mimare il comportamento di un soggetto sperimentale. La scelta dei tipi di operatori, oltre a dover rispettare necessariamente una serie di proprietà formali come *chiusura* e *sufficienza* (Frias-Martinez e Gobet, 2007; Koza, 1992; Poli et al., 2008), dipende da diversi fattori, come ad esempio il

livello di analisi che si prende in considerazione o il tipo di modello che si vuole ottenere, ad esempio modelli connessionisti o simbolici. In questo caso, prenderemo in esame una classe di semplici modelli simbolici che legge gli stimoli A e B (i terminali) e che come possibili operatori può svolgere le seguenti azioni: (i) 'compare': questo operatore compara i due stimoli e come output passa l'informazione all'operatore motorio 'respond', (ii) 'compare accumulate': a differenza di 'compare', questo operatore seleziona campioni di evidenza ogni n millisecondi e solo quando la comparazione tra campioni supera una certa soglia (definita in maniera arbitraria) una decisione viene presa in favore delle opzioni 'uguale' o 'diverso', (iii) 'respond': questo operatore effettua la risposta 'uguale' o 'diverso' e termina il trial, (iv) 'putSTM': questo operatore scrive il valore di un terminale in un buffer di memoria a breve termine, (v) 'decay': questo operatore funziona come decadimento del valore in memoria a breve termine, in proporzione alla durata della ritenzione in memoria a breve termine.

Agli operatori vengono assegnati dei tempi di esecuzione basati sui risultati della letteratura precedente (si veda Frias-Martinez e Gobet, 2007; Lane et al., 2014); inoltre, ogni operatore ha una probabilità di fallire (generalmente fissata attorno al 2%) in modo da generare un sistema non deterministico.

Come primo passo, un numero di modelli viene generato in maniera casuale o sulla base di risultati precedenti. Ad esempio, un primo modello casualmente generato potrebbe essere il primo individuo in Figura 1 che prende come input A e B, li compara ed infine risponde. Gli alberi seguenti di Figura 1 possono essere ottenuti tramite mutazione o crossover tra due individui.

La ricerca tra modelli viene fatta tramite Programmazione Genetica che cerca di ottimizzare una *funzione di fitness*, che nel nostro caso è definita come la differenza tra i dati e le predizioni dei modelli, nell'accuratezza e nei tempi di reazione. Ulteriori limiti possono essere imposti alla funzione di fitness come, ad esempio, una misura che penalizza ogni modello proporzionalmente alla grandezza/complessità delle soluzioni generate; Pirrone e Gobet (2020) hanno dimostrato, in uno studio di simulazione, che imporre limiti è importante per stimare correttamente i modelli ed evitare bloating, la tendenza delle soluzioni nel crescere a dismisura, soprattutto per cercare di spiegare il rumore inevitabilmente presente nei dati.

La Programmazione Genetica dipende da un numero di parametri che definiscono alcuni dettagli fondamentali dell'algoritmo; tra i parametri più noti (per una discussione esaustiva si veda Koza, 1992; Poli et al., 2008) riportiamo i seguenti: (i) numero di individui in ogni generazione, (ii) numero massimo di generazioni, (iii) massima profondità di un nuovo individuo, (iv) massima profondità per individui creati da crossover, (v) probabilità di un individuo di essere selezionato, proporzionalmente al valore di fitness, (vi) proporzione di individui che otterrà crossover ovunque nell'albero, (vii) proporzione di individui che otterrà crossover in un punto specifico dell'albero, (viii) massima profondità degli alberi creati da mutazione, (ix) metodo per selezionare individui nella popolazione (si veda Koza, 1992; Poli et al., 2008), (x) metodo per la generazione della popolazione iniziale (si veda Koza, 1992; Poli et al.,

2008), (xi) metodo per fermare la procedura di evoluzione (si veda Koza, 1992; Poli et al., 2008).

Il risultato finale della Programmazione Genetica, allorché la fitness viene ottimizzata completamente o il limite per fermare la procedura di evoluzione viene raggiunto, consiste in una teoria cognitiva.

Comunemente, i risultati vanno semplificati in fase di post-processing prima di essere interpretati, dal momento che i risultati possono contenere operazioni che si annullano a vicenda o un albero può essere facilmente ricondotto a una serie di operazioni più semplici, e per questo esistono svariate tecniche specifiche (Garcia-Almanza e Tsang, 2006; Javed e Gobet, 2021; Rockett, 2020) che permettono di semplificare le soluzioni ottenute.

Nell'ambito delle scienze cognitive la procedura che è stata fin qui descritta ha permesso di generare, in una serie di studi in diverse aree, interessanti spunti su come i processi di base (operatori) e i terminali interagiscono in modo da generare dei dati specifici (Addis et al., 2019; Addis et al., 2016; Frias-Martinez e Gobet, 2007; Gobet e Parker, 2005; Lane e Gobet, 2008, 2012, 2013; Lane et al., 2014; Lane et al., 2016; Sozou et al., 2017). È importante sottolineare tuttavia che finora si sono utilizzati paradigmi sperimentali particolarmente semplici in quanto il focus della ricerca è stato quello di perfezionare la metodologia GEMS piuttosto che la generazione di teorie in ambiti controversi – sebbene questo sarà il focus della ricerca futura.

3. Implicazioni teoriche

In questa ultima sezione evidenzieremo alcune implicazioni teoriche di questo approccio per la generazione di teorie scientifiche e discuteremo spunti per la ricerca futura.

Abbiamo qui presentato una metodologia che permette di generare in maniera semi-automatica teorie in scienze cognitive, utilizzando la Programmazione Genetica. Le prime applicazioni di tale approccio hanno dimostrato la sua validità e la capacità di questa tecnica nel supportare il ricercatore nell'eseguire una ricerca all'interno del vasto spazio dei modelli possibili che possono aver generato dei dati. Questa metodologia permette di ridurre il confirmation bias che spesso caratterizza la ricerca (Bilalić, McLeod e Gobet, 2010); infatti molto spesso, i ricercatori nella nostra disciplina sono interessati a confermare uno specifico modello – il più delle volte proposto da loro stessi – piuttosto che effettuare una ricerca esaustiva tra i modelli che possono aver generato dei dati.

In alcuni casi (Frias-Martinez e Gobet, 2007) le semplici teorie generate utilizzando questo approccio sono controintuitive; in questo senso, la metodologia qui descritta è naturalmente creativa e permette la scoperta di nuove teorie. Queste teorie fanno predizioni quantificabili per esperimenti futuri che possono essere testate empiricamente.

La metodologia, definita GEMS, è ancora nelle prime fasi di sviluppo e soffre dunque di semplificazioni ed aspetti controversi che devono essere sufficientemente

sviluppati. Ad esempio, per il momento le specifiche parametrizzazioni ed i tempi di esecuzione degli operatori sono fissati a valori presi dalla letteratura di riferimento (si veda ad esmpio Frias-Martinez e Gobet, 2007; Lane et al., 2016). Uno degli scopi della nostra ricerca attuale è quello di ottimizzare allo stesso tempo sia la ricerca dei modelli, come già avviene, ma anche le parametrizzazioni degli operatori. Ad esempio, l'operatore 'decay' responsabile del decadimento della traccia mnemonica, invece di essere fissato ad un valore arbitrario, può essere stimato separatamente per i diversi soggetti sperimentali sulla base dei dati generati da ogni soggetto.

Ovviamente, una teoria generata utilizzando GEMS è valida nella misura in cui gli operatori che si sono utilizzati sono validi; in questo senso, il processo di scoperta della teoria è semi-automatico, e non 'automatico', dal momento che un intervento fondamentale del ricercatore è necessario per definire gli operatori che possono generare una teoria. È chiaro che un numero elevato di operatori, soprattutto operatori che facciano predizioni qualitativamente diverse rispetto ad una specifica funzione cognitiva, sono da preferirsi rispetto ad un numero limitato di operatori o ad operatori che tendono ad avere output simili. Un ulteriore scopo della nostra ricerca è quello di raccogliere e codificare all'interno di un'architettura simbolica un numero elevato di operatori in compiti diversi come ad esempio cognizione visiva, attenzione, memoria e presa di decisioni. In questo modo, sarà possibile automatizzare anche la scelta degli operatori che verrebbero così automaticamente selezionati tra tutti quelli disponibili, tramite mutazione e crossover.

Sebbene finora questa metodologia sia stata applicata a problemi molto semplici in modo da testare la validità dell'approccio, nel futuro prossimo puntiamo ad utilizzare GEMS anche per generare teorie in compiti comparativamente più complessi che includono ad esempio, la presa di decisione ed i movimenti oculari (Krajbich, Armel e Rangel, 2010).

In conclusione, questa ambiziosa metodologia sfida l'attuale nozione di generazione di modelli scientifici: da attività puramente umana nella quale concetti astratti vengono processati per generare una teoria, ad attività ibrida uomo-computer (ma idealmente esclusivamente computerizzata) nella quale una serie di istruzioni formali vengono manipolate in modo da mimare le funzioni cognitive alla base di un compito.

Bibliografia

Addis, M., Gobet, F., Lane, P. C., Sozou, P. D. (2019). Semi-Automatic Generation of Cognitive Science Theories. In *Scientific Discovery in the Social Sciences* (pp. 155–171). Springer.

Addis, M., Sozou, P. D., Lane, P. C., Gobet, F. (2016). Computational scientific discovery and cognitive science theories. In *Computing and philosophy* (pp. 83–97). Springer.

Bilalić, M., McLeod, P., Gobet, F. (2010). The mechanism of the Einstellung (set) effect: A pervasive source of cognitive bias. *Current Directions in Psychological Science*, *19*(2), 111–115.

Chao, L. L., Haxby, J. V., Martin, A. (1999). Attribute-based neural substrates in temporal cortex for perceiving and knowing about objects. *Nature neuroscience*, *2*(10), 913–919.

Frias-Martinez, E., Gobet, F. (2007). Automatic generation of cognitive theories using genetic programming. *Minds and Machines*, *17*(3), 287–309.

Garcia-Almanza, A. L., Tsang, E. P. (2006). Simplifying decision trees learned by genetic programming. *2006 IEEE International Conference on Evolutionary Computation*, 2142–2148.

Gobet, F., Parker, A. (2005). Evolving structure-function mappings in cognitive neuroscience using genetic programming. *Swiss Journal of Psychology*, *64*, 231–239.

Javed, N., Gobet, F. (2021). On-the-fly simplification of genetic programming models. *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 464–471.

Koza, J. R. (1992). *Genetic Programming: on the programming of computers by means of natural selection*. MIT press.

Koza, J. R., Keane, M. A., Streeter, M. J. (2004). Routine automated synthesis of five patented analog circuits using genetic programming. *Soft Computing*, *8*(5), 318–324.

Krajbich, I., Armel, C., Rangel, A. (2010). Visual fixations and the computation and comparison of value in simple choice. *Nature Neuroscience*, *13*(10), 1292–1298.

Lane, P. C., Gobet, F. (2008). A methodology for developing computational implementations of scientific theories. In *Tenth International Conference on Computer Modeling and Simulation (uksim 2008)* (pp. 392–397). IEEE.

Lane, P. C., Gobet, F. (2012). A theory-driven testing methodology for developing scientific software. *Journal of Experimental & Theoretical Artificial Intelligence*, *24*(4), 421–456.

Lane, P. C., Gobet, F. (2013). Evolving non-dominated parameter sets for computational models from multiple experiments. *Journal of Artificial General Intelligence*, *4*(1), 1–30.

Lane, P. C., Sozou, P. D., Addis, M., Gobet, F. (2014). Evolving process-based models from psychological datausing genetic programming. *Proceedings of the 50th Anniversary Convention of the AISB: Computational Scientific Discovery Symposium*.

Lane, P. C., Sozou, P. D., Gobet, F., Addis, M. (2016). Analysing psychological data by evolving computational models. In *Analysis of Large and Complex Data* (pp. 587–597). Springer.

Lohn, J., Hornby, G., Linden, D. (2004). Evolutionary antenna design for a NASA spacecraft. *Genetic Programming Theory and Practice II*, 301–315.

Pirrone, A., Gobet, F. (2020). Modeling value-based decision-making policies using genetic programming: A proof-of-concept study. *Swiss Journal of Psychology*, *79*(3-4), 113.

Poli, R., Langdon, W. B., McPhee, N. F., Koza, J. R. (2008). *A field guide to genetic programming*. Lulu.com.

Rockett, P. (2020). Pruning of genetic programming trees using permutation tests. *Evolutionary Intelligence*, *13*(4), 649–661.

Simon, H. A. (1977). *Models of discovery: and other topics in the methods of science*. Dordrecht: Reidel.

Sozou, P. D., Lane, P. C., Addis, M., Gobet, F. (2017). Computational scientific discovery. In *Springer handbook of model-based science* (pp. 719–734). Springer.