Sobre a questão do Algoritmo para o Problema do Caixeiro Viajante

Ana Maria de Almeida e Maria Rosália Dinis Rodrigues

Departamento de Matemática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra
{amca, rosalia}@mat.uc.pt

Resumo: Sabemos que há problemas para os quais se demonstrou que não existem algoritmos que os resolvam (como o Problema da Paragem – Turing 1936, ou o 10º Problema de Hilbert – Matijasevic, 1970), enquanto outros há que, apesar de ser possível construir algoritmos para a sua resolução, esses algoritmos possuem "ordens de complexidade" tão grandes que originam a denominação de "ineficientes". O Problema do Caixeiro Viajante encontra-se neste último caso, sendo classificado como *NP*-completo. Mas o que significa "*NP*-completo"? E será ou não possível encontrar um algoritmo "eficiente" para este problema? De facto, não basta identificar um problema e garantir que existe pelo menos uma solução para ele, é ainda necessário saber se ele pode ser efectivamente resolvido em tempo e espaço de memória finitos.

1 Prólogo

A propósito do artigo "O Problema do Caixeiro Viajante"[2], publicado na Folha Informativa nº8 – Out/98, e a pedido de alguns colegas, serve este como resposta à questão da existência (ou não) de algoritmo para o Problema do Caixeiro Viajante. Para além disso, e no sentido da divulgação pretendida pela folha informativa abranger o máximo de público, tentámos tornar este artigo o mais informal possível. A quem pretender aprofundar os conceitos e teorias aqui aflorados recomendamos as referências [5], [6] ou [1].

2 Introdução

"– [...] Imagina que viajas para a América e que tens lá 25 amigos. Cada um deles vive numa cidade diferente e tu queres visitá-los a todos. Agarras no mapa e pensas na melhor maneira de o fazer, ou seja, o mínimo de quilómetros possível de modo a poupares o máximo de tempo e gasolina. Qual é o caminho mais curto? Como poderás descobri-lo?

[...]

– Onde está o problema insolúvel? Só preciso de calcular quantos percursos

existem e escolher o mais curto.
– Ahá! – gritou o velhote – Se fosse assim tão simples! Mas com 25 amigos já tens 25! possibilidades, e isto é um número horrorosamente grande. Mais ou menos

1600 000 000 000 000 000 000 000 00

É impossível experimentá-las todas para se saber qual é a mais curta. Nunca chegarias ao fim, mesmo com o mais potente computador que exista.
– Por isso, e numa palavra, não é possível.
– Isso depende muito. Já pensámos muito sobre este assunto. Os Diabos dos Números mais inteligentes já experimentaram com todos os truques possíveis e imaginários, e chegaram à conclusão de que às vezes resulta e outras não.
– Que pena. – Disse o Roberto – Se resulta apenas às vezes, ficamos a meio do caminho.
– E, o que é pior ainda, não conseguimos provar, definitivamente, que não existe uma solução perfeita. Nesse caso não teríamos que continuar a procurar. Teríamos ao menos demonstrado que não existe demonstração, o que, afinal de contas, seria também uma demonstração." – "O Diabo dos Números", Enzensberger.

3 Problemas, algoritmos e complexidade

Vamos começar pelo princípio, isto é, vamos definir de um modo mais rigoroso os termos utilizados no título desta secção.

Informalmente, dizemos que um *problema* não é mais do que aquilo que o algoritmo resolve mas, na verdade, é um objecto com o qual se pode trabalhar, e que podemos descrever como sendo constituído por um domínio – contendo todas as suas possíveis *concretizações* ou *particularizações* – e por uma questão geral a todas essas concretizações (no sentido em que pode ser aplicada a qualquer uma delas) produzindo uma *resposta*. Formalmente, podemos estabelecer um problema usando um par do seu domínio (C, R), onde C é uma descrição genérica de uma qualquer concretização, e R é uma *resposta*. O Problema do Caixeiro Viajante – **PCV** – aparece então formalizado como:

Caixeiro Viajante
Concretização: Conjunto de cidades,
$$c_1, \ldots, c_n,$$
e distâncias inteiras positivas[3], $d(c_i, c_j)$, com $i, j = 1, \ldots, n$.
Resposta: Permutação π do conjunto dos índices que minimiza
$$d(c_{\pi(n)}, c_{\pi(1)}) + \sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}).$$

Um *algoritmo* é um método faseado de resolução de um dado problema, no sentido em que, quando aplicado a uma qualquer particularização do problema, *garante* a obtenção de uma solução. Podemos formalizar um algoritmo (ou método de resolução) de diferentes modos, nomeadamente, através de Máquinas de Turing ou de programas de computador, mas, e principalmente neste último caso, quando se pretende utilizar uma máquina para tratar problemas reais, procuramos encontrar algoritmos que, não só garantam a obtenção da solução desejada, mas que resolvam de um modo "eficiente" o problema proposto. Mas o que é um algori-

[3]Sem perda de generalidade pois, num computador, os irracionais são aproximados por racionais que podem, por multiplicação por um factor apropriado, ser transformados em inteiros.

tmo "eficiente"? Em geral, a resposta a esta pergunta, e apesar de haver outros recursos envolvidos, relaciona eficiência com tempo: procuramos construir algoritmos capazes de solucionar problemas *em tempo útil*.

Um problema só está *resolvido* se for possível construir um método geral para resolver qualquer concretização desse problema embora, na prática, só se possa resolver um conjunto finito de concretizações (até um dado comprimento máximo) devido a limites físicos dos recursos disponíveis. No entanto, se os métodos são gerais, poderão ser aplicados à resolução de concretizações maiores caso os recursos possam ser aumentados. Para o problema aqui em causa, o *tamanho* de cada concretização será, necessariamente, o número n de cidades e parece óbvio afirmar que o tamanho da concretização afecta o comportamento dos métodos de resolução em termos de utilização de recursos pois, quanto mais cidades, mais distâncias e, muito provavelmente, mais tempo gasto. Mas como podemos efectivamente estimar o tempo de resolução de um algoritmo quando aplicado a uma qualquer concretização?

Essa estimativa permite medir a *complexidade* de um método ou algoritmo, e é feita através da determinação de um *limite superior* para a quantidade do recurso – tempo – gasta pela aplicação do método de resolução a concretizações de tamanho n, e que é usualmente designada, em Ciências da Computação, como o *pior dos casos*[4], isto é, o pior que se pode esperar quando um dado método ou procedimento usa um dado recurso. Somos, então, conduzidos à definição seguinte:

> Um método diz-se *da ordem de $F(n)$*, e escreve-se $\mathcal{O}(F(n))$, se existir uma constante k tal que o tempo de execução do método, para qualquer concretização de tamanho n, é limitado superiormente por $kF(n)$.

F(n)	n=10	n=20	n=30	n=40	n=50
n	0.00001 segundos	0.00002 segundos	0.00003 segundos	0.00004 segundos	0.00005 segundos
n^2	0.0001 segundos	0.0004 segundos	0.0009 segundos	0.0016 segundos	0.0025 segundos
n^3	0.001 segundos	0.008 segundos	0.027 segundos	0.064 segundos	0.125 segundos
n^5	0.1 segundos	3.2 segundos	24.3 segundos	1.7 minutos	5.2 minutos
2^n	0.001 segundos	1.0 segundos	17.9 minutos	12.7 dias	35.7 anos
3^n	0.59 segundos	58 minutos	6.5 anos	3855 séculos	$2x10^8$ séculos

Tabela 1 Comparações entre alguns tipos de ordens de complexidade relativamente ao recurso tempo.

[4]Pode perfeitamente acontecer que um dado método, para alguma concretização (ou mesmo para a maioria delas), utilize o recurso em causa em muito menor quantidade que a assim calculada. No entanto, a garantia mais rigorosa para concluir sobre a complexidade de um dado método relativamente a qualquer uma das suas concretizações encontra-se neste limite superior.

Assim sendo, um método dir-se-á, por exemplo, *Linear* se for $\mathcal{O}(n)$, *Polinomial* se for $\mathcal{O}(n^p)$ ou *Exponencial* se for $\mathcal{O}(p^n)$.

Claro que a "qualidade" real de classificações como as anteriormente apresentadas vai depender do tipo de máquinas disponíveis para a aplicação do método em causa. Se, por exemplo, usarmos uma máquina imaginária (e nem tão imaginária como isso) que, em termos de tempos de execução, apresenta os valores da tabela 1, podemos ver como a utilização do tempo pode variar em função do tamanho n, para alguns exemplos típicos de ordens de complexidade e para concretizações relativamente pequenas. Facilmente se verifica que, mesmo que algumas exponenciais comecem por apresentar valores mais baixos que algumas polinomiais, muito cedo perdem essa vantagem para passarem a apresentar valores incríveis (mas totalmente correctos)! Claro que, com uma máquina mais rápida, os tempos apresentados sofreriam um certo decréscimo mas este decremento só é realmente notório em métodos polinomiais pois, para os exponenciais, a ordem de complexidade é tão grande que a resolução se torna impraticável mesmo para concretizações de dimensões relativamente pequenas. De facto, suponha-se que temos 2 métodos, um com ordem de complexidade $\mathcal{O}(n^3)$ e outro da $\mathcal{O}(2^n)$, e uma máquina suficientemente rápida para que ambos resolvam concretizações de tamanho $n = 100$ em *uma hora*. Se conseguirmos um computador duas vezes mais rápido, podemos agora resolver com o método polinomial, na mesma hora, particularizações com $n = 126$ enquanto que o método exponencial só vai resolver, no mesmo tempo, concretizações com tamanho $n = 101$...

4 Métodos polinomiais e problemas de decisão – A classe P

Pelo pequeno exemplo apresentado na tabela 1 percebe-se que é desejável conseguir métodos *quando muito* polinomiais para aplicar à resolução de problemas. De facto, a maioria dos métodos exponenciais não é mais que uma variante do método de Procura Exaustiva – procura-se, entre *todas* as soluções possíveis para uma dada concretização de um problema, aquela que melhor satisfaz os requisitos exigidos. Por outro lado, métodos polinomiais aparecem, em geral, devido à apreensão de propriedades intrínsecas à estrutura do problema, permitindo reduzir o conjunto de soluções possíveis para aquelas ditas *admissíveis* ou mesmo a uma *única* solução bem caracterizada.

A maioria dos investigadores concorda que um problema só se diz *completamente resolvido* quando encontrado um método polinomial que o resolva e, mais ainda, diz-se que um problema *não pode ser tratado* (do inglês *intractable*) caso seja demasiadamente complexo para que possa existir um método polinomial que o resolva. Esta procura da classificação polinomial de problemas é muito importante pois permite tirar conclusões, com grande exactidão, sobre o comportamento de um método relativamente ao uso de um dado recurso o que, como vimos, já não acontece nos exponenciais[5]. O próximo passo torna-se óbvio: decidir, *por análise do problema*, se pode ou não existir

[5]Mais ainda, a complexidade de um problema pode ser considerada, no seu essencial, como independente da representação escolhida desde que esta seja uma codificação "sensata" da concretização em causa. Para mais promenores ver [5].

um algoritmo polinomial que o resolva! Entramos no domínio do estudo da *Complexidade de Problemas* ao invés da complexidade de algoritmos. Restringimos, aqui o nosso estudo aos, assim denominados, *Problemas de Decisão*, que não são mais do que problemas cuja resposta é "Sim" ou "Não". O Problema do Caixeiro Viajante não é um problema de decisão pois a resposta de cada concretização é a permutação que minimiza a distância. Pode, no entanto, ser facilmente reformulado como um problema de decisão, **PCV–Dec**:

Caixeiro Viajante–Decisão

Concretização: Conjunto de cidades, $c_1, \ldots, c_n$, distâncias inteiras positivas, $d(c_i, c_j)$, com $i, j = 1, \ldots, n$ e $i \neq j$, e um inteiro positivo k.

Resposta: "Sim", se e só se existe uma permutação π do conjunto dos índices tal que

$$d(c_{\pi(n)}, c_{\pi(1)}) + \sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) \leq k \, .$$

Todos os problemas de decisão para os quais ou se conhecem ou se pode, comprovadamente, construir métodos polinomiais de resolução são agrupados sobre a definição geral de problemas pertencentes à classe P[6]. Podemos portanto dizer que, se um problema de decisão não pertence a P, é um problema *difícil*. Mas qual a relação entre o PCV–Dec e o PCV? Será que, se um pertencer a P, o outro também pertencerá, ou não podemos relacioná-los deste modo? [7] mostra que [7],

Teorema 1 *Existe um algoritmo polinomial em tempo para o PCV* **sse** *existe um algoritmo polinomial em tempo para o PCV–Dec.*

Este teorema garante-nos que podemos, de facto, restringir a nossa atenção ao problema PCV–Dec e procurar saber se este é ou não um problema difícil, procurando responder à seguinte questão: será que PCV–Dec pertence a P?

Sabemos que existem problemas que não estão em P: se o problema não é resolúvel por nenhum algoritmo, então, não pode pertencer a P, como no caso do Problema da Paragem de Turing[8]. De facto, Alan Turing demonstrou, no primeiro terço do século XX, que existem problemas *não decidíveis* (do inglês, *undecidable*) pois são tão complicados que não existe nenhum algoritmo que os possa resolver. Estes problemas constituem o exemplo perfeito de um problema que não pode ser tratado (podemos dizer que são "intratáveis")! Por outro lado, existem problemas que só são resolúveis por algoritmos exponenciais e que, portanto, também não pertencem a P. No entanto, e apesar de todos os indícios sugerirem que a resposta para a questão do PCV–Dec estar em P é negativa, não nos será possível apresentar provas concretas nesse sentido, pois, *para já*, elas não existem.

[6] Seguindo a notação de [6] esta classe está contida na classe mais geral FP onde se agrupam todos os problemas com algoritmos polinomiais sem restrições .

[7] Resultados análogos podem ser obtidos para a maioria dos problemas de optimização [5].

[8] Não é possível construir um algoritmo que, para um qualquer programa de computador e dados arbitrários, consiga decidir se esse programa, quando aplicado a esses dados, pára ou continua indefinidamente.

5 Não Determinismo – A classe *NP* e problemas completos para *NP*

Até ao momento, não conhecemos nenhum método que resolva o PCV–Dec em tempo polinomial. De facto, existe um número factorial de caminhos (permutações) possíveis de definir que poderão ou não ser menores do que k. No entanto, dado um qualquer caminho particular, I, podemos provar que a resposta associada é "Sim"(S) bastando para isso somar as distâncias correspondentes e comparar com k, sendo esta verificação obviamente feita em tempo polinomial no comprimento de I. Isto não implica que o problema possa ser resolvido em tempo polinomial pois não procurámos entre as várias permutações existentes mas, apenas, se uma certa permutação verifica a propriedade desejada.

A classe *NP* pretende incluir este tipo de problemas, ou seja, problemas de decisão para os quais se pode verificar se uma dada concretização corresponde a uma resposta S em tempo polinomial. No sentido de podermos classificar estes problemas como pertencentes à classe *NP*, podemos construir um método, dito *não–determinista* que "resolva" um dado problema X, no sentido de verificar todas as respostas S, em duas fases:

1. Dada uma qualquer concretização x do problema, obter, de algum modo não–determinista ("adivinhar"), uma sequência y tal que $|y| \leq p(|x|)$ [9], para algum polinómio p;

2. Executar um algoritmo de verificação em x e y que devolva a resposta "sim" ou "não".

Este *método não–determinista* é uma ferramenta poderosa: podemos descrevê-lo como um algoritmo que possui uma instrução especial que provoca subdivisões na computação, que passa assim a decorrer em paralelo. Como é evidente, estas computações paralelas podem crescer exponencialmente no tempo decorrido, como ilustra a figura 1.

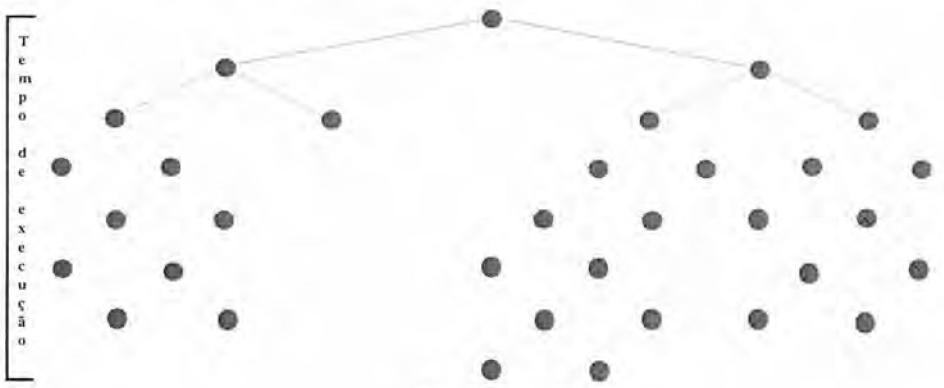

Figura 1

[9] $|x|$ representa o tamanho de x.

Estas considerações levam-nos à seguinte definição : um problema de decisão pertence à classe **NP**[10] se pode ser resolvido por um método não–determinista com um limite polinomial no recurso tempo.

Facilmente se verifica que o PCV–Dec está em *NP*. Mas que relação existe entre *P* e *NP*? Obviamente, $P \subseteq NP$ pois um algoritmo determinista polinomial não é mais do que um caso particular de um método não–determinista polinomial que não usa a instrução de paralelismo. Parece, no entanto, difícil acreditar que possa existir um algoritmo puramente determinista que possa simular não–determinista usando, *apenas*, um gasto polinomial de tempo.

Por um lado, é óbvio que, se $P = NP$, o PCV–Dec estaria em *P* e poderíamos começar a procurar um algoritmo polinomial para o PCV, com a certeza que ele existiria. Por outro lado, e apesar de os investigadores do domínio da Teoria da Complexidade acreditarem na hipótese $P \neq NP$, não foi até agora encontrada qualquer prova para a conjectura de que *P* esteja propriamente contido em *NP*. Quer isto dizer que não podemos afirmar que o PCV–Dec não pode ser resolvido em tempo polinomial a menos que se prove que $P \neq NP$.

Prova-se, no entanto, que,[11]

$$P \neq NP \Longleftrightarrow \text{PCV–Dec} \notin P \ .$$

O problema *X* diz-se *transformável* para *Y* se existe uma transformação constructiva que aplica qualquer concretização do problema *X* para uma concretização equivalente do problema *Y*. Note-se que também uma transformação deve obedecer a limites para os recursos que utiliza, donde, se o tempo usado for polinomialmente limitado, podemos dizer que *X* é *polinomialmente transformável* para *Y*[12].

Define-se um problema *X* como polinomialmente *difícil* para *NP* se $\forall Y \in NP$, *Y* é polinomialmente transformável para *X*. E diz-se que *X* é *completo para NP* se *X* é polinomialmente difícil para *NP* e $X \in NP$.

Quer isto dizer que, *X* é *NP–completo* sse:

- $X \in NP$;
- qualquer outro problema em *NP* é polinomialmente transformável para *X*.

Existem vários problemas já identificados como *NP*–completos, sendo o da *Exequibilidade*[13] um dos primeiros:

Exequibilidade

Concretização: Conjunto de literais $U = \{u_1, \bar{u}_1, \ldots, u_n, \bar{u}_n\}$ e conjunto de claúsulas $C = \{c_1, \ldots, c_m\}$ onde cada c_i , $1 \leq i \leq m$ é constituído por literais de *U* (está na forma conjuntiva normal).

Resposta: "Sim", se e só se existe uma atribuição de valores lógicos a cada um dos literais em *U* tal que todas as claúsulas de *C* são satisfeitas (isto é, são verdadeiras).

[10] Cook e Karp foram os primeiros a mencionar e definir esta classe de problemas e foi Karp quem, em 1972, apresentou a denominação *Nondeterministic Polynomial time–NP*.

[11] Ou, equivalentemente, que o PCV–Dec é *NP*–completo.

[12] Para a formalização de polinomialmente transformável ver [1] ou [6]

[13] Tradução livre do inglês *Satisfiability* . O resultado da identificação deste problema como NP–completo é o denominado *Teorema de Cook*.

Outro exemplo de um problema *NP*–completo que voltaremos a encontrar (e que muito nos vai ajudar) é o do

Ciclo Hamiltoniano – **CH**
Instância: Grafo, $G = (\mathcal{V}, \mathcal{A})$.
Resposta: "Sim", se e só se existe um caminho fechado que inclui todos os nós de G.

Dizer que um problema X é *NP*–completo, pode ser interpretado como dizer que X é um dos problemas mais difíceis em *NP*, segundo transformações compatíveis com P. Quer isto dizer que, se o PCV–Dec for *NP*–completo e se estiver em P, então, $P = NP$. Assim, é de todo o interesse saber se um problema é completo para *NP* segundo reduções compatíveis com P pois, então, pode afirmar-se que esse problema será resolvido em tempo polinomial se e só se $P = NP$.

6 PCV–Dec é *NP*–completo

Como vimos, a complexidade de um problema *NP*–completo está intimamente relacionada com a conjectura $P \neq NP$. Daí que, usualmente, a demonstração de que um novo problema é *NP*–completo ultrapasse a mera prova imediata e seja um exercício de (tentativa) de demonstração da conjectura: se conseguirmos provar que um problema completo para *NP* possui um algoritmo polinomial ...

No caso que temos em mãos, nada disto vai acontecer. Vamos mostrar apenas uma das provas típicas de *NP*–completude para o PCV-Dec, que ajuda também a exemplificar uma das técnicas de transformação entre problemas.

Note-se, em primeiro lugar, que as transformações (polinomiais ou não), são transitivas pelo que, se conseguirmos transformar um problema conhecido como *NP*–completo para o PCV-Dec, então todos os outros problemas em *NP* são também transformáveis para o PCV-Dec e, como este está em *NP*, está provado o pretendido.

Mais ainda, vamos provar que o PCV é *NP*–completo, provando que ele contém um caso particular (uma restrição) que é *NP*–completo. Nesse sentido, observemos o seguinte problema:

Caixeiro Viajante–Simétrico
Concretização: Conjunto de cidades, $c_1, \ldots, c_n$, distâncias inteiras positivas, $d(c_i, c_j)$, com $i, j = 1, \ldots, n$, $i \neq j$ e tais que $d(c_i, c_j) = d(c_j, c_i)$, e um inteiro positivo k.
Resposta: "Sim", se e só se existe uma permutação π do conjunto dos índices tal que

$$d(c_{\pi(n)}, c_{\pi(1)}) + \sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) \leq k \, .$$

Apesar de este ser o problema em que, habitualmente, as pessoas pensam quando se referem ao Caixeiro Viajante, ele é, de facto, um caso especial do problema mais genérico (note-se que tivemos de impôr uma restrição aos valores possíveis para as distâncias entre as cidades). Podemos ainda especializar mais, impondo a restrição adicional $d(c_i, c_j) + d(c_j, c_k) \geq d(c_i, c_k)$, isto é, impomos uma desigualdade triangular. É evidente que o Caixeiro Viajante, simétrico e com desigualdade triangular é um caso particular do problema inicial. Se este for *NP*–completo, por força maior, também o será o PCV.

Seja $G = (\mathcal{V}, \mathcal{A})$ um grafo qualquer. Defina-se conjunto de cidades,

$c_1, \ldots, c_n$, como sendo constituído por todos os nós de V e ainda, para i,j =1, ... , n ,

$$d(c_i, c_j) = \begin{cases} 1 & \text{se } c_i \text{ liga a } c_j \\ 2 & \text{caso contrário} \end{cases} .$$

Assim sendo, temos que

$$d(c_i, c_j) = d(c_j, c_i) \quad \text{e}$$

$$d(c_i, c_j) + d(c_j, c_k) \geq d(c_i, c_k) .$$

Ora, é evidente que existe um ciclo hamiltoniano se e só se existe um caminho fechado com distância $\sharp V$ e construímos, assim, para uma qualquer concretização do problema do ciclo hamiltoniano, uma concretização equivalente do PCV, simétrico e com desigualdade triangular. Mais ainda, o ciclo hamiltoniano é, necessariamente, o caminho mais curto entre todos os vértices, o que significa que, se resolvermos o CH, também resolvemos o PCV. Como, de acordo com o anteriormente exposto, o CH é *NP*–completo, está concluída a nossa demonstração de que

Teorema 2 *O PCV–simétrico e com desigualdade triangular é NP–completo.*

E, como este é, apenas, uma restrição do PCV geral, concluímos finalmente que:

O PCV–Dec é *NP*–completo.

7 Problemas, complexidade e aproximações

Mesmo que um problema qualquer seja classificado como *NP*–completo, se foi suficientemente importante para ser estudado é porque, provavelmente, é suficientemente importante para que seja necessário encontrar uma solução (ou resolução). Uma das consequências práticas da classificação como *NP*–completo (ou difícil) é a da restrição de estratégias e concentração de energias numa das opções seguintes:

Comecemos por notar que, em muitos casos, apenas um pequeno detalhe na descrição de um problema pode provocar uma mudança de P para *NP*–completo. Assim, e dado um qualquer problema, este não deve ser encarado taxativamente, isto é, embora por um lado possa intuitivamente parecer que deve ser possível provar que é *NP*–completo, deve-se, por outro lado, tentar descobrir se é possível construir um algoritmo que o resolva em tempo polinomial ou vice-versa.

Mesmo quando é possível demonstrar que um dado problema é completo para *NP*, e caso a necessidade de obter uma solução exacta seja premente (e o risco de uma grande perda de tempo seja secundário), nem tudo está perdido: o facto de o problema geral X ser completo para *NP* não implica necessariamente que a mesma classificação se imponha para todos os seus subproblemas ou restrições. Esses subproblemas, sendo particularizações, são independentes em termos de classificação pois que, enquanto alguns sejam ainda *NP*–completos, outros haverá que podem ser resolvidos em tempo polinomial, como é o caso da Compactação de Elementos Rectangulares numa área Rectangular, problema que está classificado como NP-completo mas que, no entanto, possui subproblemas demonstradamente em P (ver [1]). Uma análise mais profunda do problema geral poderá, então, possibilitar a identificação de casos particulares que sejam suficientemente importantes para serem salientados e estudados de per si, o que, even-

tualmente, poderá conduzir à descoberta de problemas particulares resolúveis em tempo polinomial. Poderá mesmo descobrir-se que as concretizações para as quais o problema não pode ser resolvido por um algoritmo polinomial são raras e possíveis de identificar a priori.

Por outro lado, e porque o tempo é, na maioria dos casos, um bem precioso, poderá ser possível "aproximar" a resolução do problema através do uso de algoritmos polinomiais, algoritmos esses que **não garantem** a obtenção da solução que satisfaz totalmente a pergunta feita (a "melhor" solução ou *solução óptima*) mas garantem, pelo menos, a obtenção de uma resposta que a satisfaz parcialmente (aproxima a solução óptima). Isto porque, dependendo do contexto em que o problema se põe e, caso não seja humanamente possível resolver o problema devido à relativa efemeridade da vida humana, poderá ser importante obter soluções "parciais" para a sua resolução. Neste caso, os algoritmos que aproximam a melhor solução são designados de *heurísticas*, e são especialmente importantes na generalidade dos problemas que surgem na área da Optimização Combinatória. Nestes casos, e o PCV é um deles, temos problemas de procura de extremos com três constituintes fundamentais [5]:

- um conjunto de concretizações, C;
- um conjunto finito de *soluções admissíveis*, $S(I)$, para cada concretização $I \in C$;
- uma função f, dita *função objectivo* que faz corresponder a cada par $(I, s) \in C \times S(I)$ um valor numérico denominado *valor da solução* s.

Um extremo absoluto para a concretização I é dito *solução óptima* e será este uma solução admissível $s^\star \in S(I)$ tal que $\forall s \in S(I), f(I, s^\star) = extr\{f(I, s)\}$, onde *extr* designará *mínimo* ou *máximo*, consoante for o caso.

Caso o problema seja completo (ou difícil) para *NP*, o melhor que podemos esperar conseguir em tempo polinomial é a obtenção de um extremo local ao invés do esperado extremo global, o que pode ser conseguido através do uso de heurísticas. Evidentemente que, a construção de heurísticas tenderá a procurar métodos *quando muito* polinomiais, caso contrário, estaríamos de volta ao ponto de partida.

Referências

[1] Almeida, A. M. de: *Uma Abordagem Particular ao estudo de alguns Problemas NP-completos*, Departamento de Matemática, FCTUC, 1994.

[2] Batel Anjo, A.J., Rodrigues, M.R.D.: *O Problema do Caixeiro Viajante*, Folha Informativa nº8 do Boletim da Sociedade Portuguesa de Matemática, Outubro, 1998.

[3] Edmonds, J.: *Paths, trees and flowers*, Canad. J. Math., 17, 1965, pp. 449–467.

[4] Enzensberger, H.M.: *O Diabo dos Números*, Edições ASA 1998.

[5] Garey, M.R., Johnson, D.S.: *Computers and Intractability – A Guide to the Theory of NP-Completeness*, Bell Laboratories, Murray Hill, New Jersey, 1979.

[6] Johnson, D.S.: *Algorithms and Complexity – A Catalog of Complexity Classes*, Handbook of Theoretical Computer Science, Editor Jan Van Leeuwen, Vol.A, Elsevier Science Publishers B.V., 1990, pp. 67–161.

[7] Johnson, D.S. e Papadimitriou, C.H.: *Computational complexity, The Traveling Salesman Problem*, Edited by E.L.Lawler, J.K.Lenstra, A.H.G.Rinnooy Kan, D.B.Shmoys, John Wiley & Sons Ltd., 1985, pp. 37–85.

[8] Turing, A.: *On computable numbers, with an application to the Entscheidungsproblem*, Proc. London Math. Soc., Serie 2, 42, 230–265 e 43, 544–546, 1936.