

Universidade Aberta



Gerador de redes Actividades nos Arcos

Raimundo Jorge dos Santos Baptista Ferreira

Orientador: Professor Doutor José Pedro Fernandes Silva Coelho

Trabalho Final
Licenciatura em Informática

2008

Agradecimentos

Ao meu orientador, Professor Doutor José Coelho, pela oportunidade concedida e toda a disponibilidade e empenho que sempre demonstrou.

A todos os meus amigos e colegas que me apoiaram, em especial ao António Vieira.

Finalmente um agradecimento especial à minha esposa Lurdes por todo o apoio, sacrifício e paciência.

Índice

1	Enquadramento.....	1
1.1	Introdução	1
1.2	Objectivo.....	2
1.3	Definições	3
1.3.1	Projecto	3
1.3.2	Actividade.....	3
1.3.3	Recursos	3
1.3.4	Precedências	3
1.3.5	Representação de actividades nos arcos	5
1.3.6	Engine Tester.....	8
2	Gerador aleatório de actividades	9
2.1	Principais características do gerador.....	9
2.2	Geração da rede	9
2.2.1	Método de geração por remoção	10
2.2.2	Método de geração por adição	11
2.2.3	Escolha do método de geração	12
2.3	Geração de parâmetros.....	13
2.3.1	Geração automática de valores	13
3	Problemas detectados	15
3.1	Falta de arcos livres na geração de redes por remoção	15
3.2	Não são gerados todos os projectos possíveis	16
3.3	Geração de lista de precedências com maior probabilidade que outras.....	17
3.4	Geração do número de arcos.....	19
4	Implementação	21
4.1	Diagrama de classes	21
4.2	Gerador Projecto	22
4.3	Gerador de rede por remoção.....	23

4.4	Gerador de rede por adição	25
5	Testes.....	27
5.1	Testes de geração aleatória	27
5.2	Testes de velocidade	28
5.3	Testes variação do número de nós mantendo o número de arcos	31
6	Conclusões	33
	Referências	34
	Acrónimos	34
	Apêndices I - Manual de utilizador	35
	Apêndices II – Exemplo instância saída.....	39
	Apêndices III – Classes principais.....	40

Índice de Figuras

Figura 1 – rede em que as actividades são todas feitas em sequência.....	1
Figura 2 - representação actividades nos nós	4
Figura 3 - representação actividades nos arcos	5
Figura 4 - actividades nos arcos com actividades fictícias	6
Figura 5 - actividades nos arcos com actividades fictícias para representar actividades paralelas	6
Figura 6 - actividades nos arcos com actividades fictícias para só ter um nó de início e um nó final.....	6
Figura 7 – matriz triangular superior de uma rede com 5 nós	7
Figura 8 - matriz de adjacências completa para uma rede com 5 nós	10
Figura 9 - matriz demonstração do problema do método de remoção	15
Figura 10- frequência relativa acumulada do número de ciclos sem operação de adição ou remoção	16
Figura 11 - problema na adição de actividades fictícias 1	16
Figura 12 - problema na adição de actividades fictícias 2.....	17
Figura 13 - todas matrizes de uma rede 5 nós e 5 arcos	18
Figura 14 - diagrama de actividades nos nós equivalentes.....	19
Figura 15 - número de arcos em função do número de nós.....	20
Figura 16 - diagrama das principais classes	21
Figura 17 - geração projectos	22
Figura 18 - geração de rede por remoção	23
Figura 19 - geração rede por adição	25
Figura 20 - variação de nº nós / nº arcos / nº recursos	27
Figura 21 - tempo de execução do algoritmo pela diferença do nº arcos - nº nós.....	28
Figura 22 - variação de nº arcos / tempo / nº recursos.....	29
Figura 23 - gráfico de análise de tendência do tempo de execução	29
Figura 24 - tempo execução / nº arcos com grupos no ° de recursos.....	30
Figura 25 - tempo / nº arcos.....	31
Figura 26 - tempo execução / nº nós.....	32
Figura 27 -gráfico de análise de tendência do tempo de execução	32

Índice de Equações

Equação 1 - calculo do número máximo de arcos	7
Equação 2 - geração do nó origem do arco[1]	10
Equação 3 - geração nó de destino do arco	10
Equação 4 - número arcos remover a remover no método remoção	11
Equação 5 - número arcos livres.....	11
Equação 6 - geração nó de origem do arco.....	11
Equação 7 - número de arcos a remover após aplicação método de adição	12
Equação 8 - número de arcos a gerar em que a quantidade de operações é igual em ambos os métodos.....	12
Equação 9 - média adaptada para geração do número de arcos[1].....	20
Equação 10 - desvio padrão para geração do número de arcos[1]	20
Equação 11 - geração do número de arcos implementada.....	20

Índice de Tabelas

Tabela 1 - número de redes possíveis dado o número de arcos e número de nós[1].....	19
---	----

1 Enquadramento

1.1 Introdução

A realização de testes de software, apesar de muitas vezes ser descurada, é fundamental para a detecção precoce de possíveis erros, que quando detectados em fase posterior poderiam ter consequências catastróficas. A realização de software perfeito, isto é sem erros, é uma utopia. Isto é, muito provável que existam erros nas várias fases do desenvolvimento de software.

Para o desenvolvimento, ou aperfeiçoamento, de novos métodos de resolução de problemas de calendarização de projectos é necessário criar baterias de testes. Estes podem ser efectuadas utilizando problemas:

- Criados para o efeito;
- Problemas conhecidos, de que é exemplo o “*Project Scheduling Problem Library*” PSPLib¹;
- Geradores.

Os geradores são a possibilidade mais flexível, pois permitem criar um grande número de problemas diferentes, em pouco tempo. Têm no entanto o senão de poderem gerar instâncias sem grande utilidade em termos de testes. Na rede da Figura 1, todas as suas actividades são executadas em sequência, sendo por isso um exemplo de uma rede com pouca utilidade.

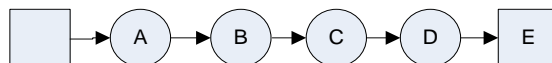


Figura 1 – rede em que as actividades são todas feitas em sequência

¹ Problemas PSPLIB podem ser descarregados em: <http://129.187.106.231/psplib/main.html>

1.2 Objectivo

O presente trabalho tem como objectivo a implementação do algoritmo apresentado em “A *Random Activity Network Generator*”[1] que permite gerar redes actividades nos arcos, seguindo a filosofia de *Engine Tester* (ver [2]).

1.3 Definições

1.3.1 Projecto

Um projecto é um trabalho de duração finita para se atingir determinado objectivo.

Para tornar exequível a gestão de um projecto este deverá ser dividido em actividades e quando necessário estas também poderão ser divididas.

1.3.2 Actividade

Uma actividade é uma parte do trabalho necessário efectuar para se concluir o projecto.

Numa actividade deverá ser possível estimar o tempo e os recursos necessários para a concluir.

1.3.3 Recursos

Para a realização de projectos são necessários recursos que podem ser de três tipos:

- Renováveis: recursos que não são limitados em número ao longo do tempo mas são limitados num instante temporal;
- Não renováveis: recursos que tem quantidade finita para todo o projecto;
- Duplamente constrangidos: recursos limitados num instante temporal, que têm uma quantidade finita para todo o projecto e podem ser cobertos por recursos renováveis e não renováveis[3].

1.3.4 Precedências

Quando há uma precedência da actividade A para actividade B para se iniciar a actividade B é necessário que actividade A tenha terminado.

Normalmente uma actividade inicia logo após todas as que lhe precedem terminarem mas pode ser necessário um tempo de espera.

Exemplo: Na construção após colocar o cimento tem que se deixar secar antes de ser possível pintar.

Usualmente uma actividade está dependente do fim de outra para iniciar, mas existem quatro tipos de precedências[4]:

- Fim Início: o início de uma actividade está dependente do fim de outra;
- Fim Fim: o fim de uma actividade está dependente do fim de outra;
- Início Fim: o fim de uma actividade está dependente do início de outra;
- Início Início: o início de uma actividade está dependente do início de outra.

Existem duas representações de precedências entre actividades utilizando grafos sem ciclos[4]:

- Actividades nos nós (AoN);
- Actividades nos arcos (AoA).

Na representação de actividades nos nós, as actividades são representadas pelos nós e as precedências pelos arcos.

Na Figura 2 é uma representação de actividades nos nós da seguinte lista de tarefas²: A; B; C(A); D(B);E(C,D).

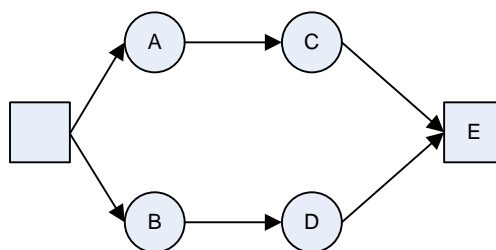


Figura 2 - representação actividades nos nós

Na representação de actividades nos arcos, as actividades são representadas pelos arcos e os nós representam eventos. As precedências estão implícitas.

Na representação de actividades nos arcos um projecto pode ter várias representações.

² E(C,D) – representa a actividade E tem como precedentes as actividades C e D

A Figura 3 tem a mesma lista de precedências da Figura 2 mas numa representação de actividades nos arcos.

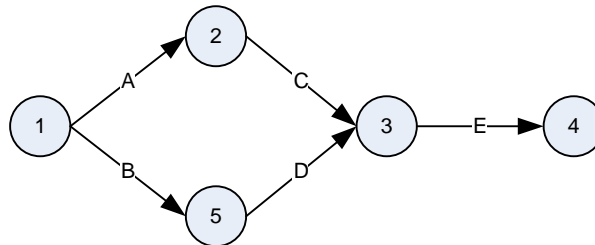


Figura 3 - representação actividades nos arcos

1.3.5 Representação de actividades nos arcos

1.3.5.1 Características de actividades nos arcos

Características de uma representação de actividades nos arcos[5]:

1. Uma actividade só pode começar quando, todas as actividades que incidem no nó de onde a actividade emerge tiverem terminado;
2. O comprimento dos arcos e o seu ângulo não têm qualquer significado;
3. A identificação de cada nó é única e para evitar ciclos o nó de destino de um arco tem índice superior ao do nó de origem;
4. Entre dois nós não pode existir mais que um arco;
5. O grafo não pode conter ciclos;
6. A rede só tem um nó de início, nó sem arcos a incidir, e só um nó de fim, nó sem arcos a emergir.

1.3.5.2 Actividades fictícias

Para poder representar todas listas de precedências, a representação de actividades nos arcos, necessita de actividades fictícia, isto é, arcos que não são actividades mas existem para preservar as precedências.

Motivos que podem justificar a utilização de actividades fictícias[5]:

1. Para poder-se representar algumas estruturas;

A lista de precedências: A; B; C(A); D(A,B) poderá ter a representação da Figura 4.

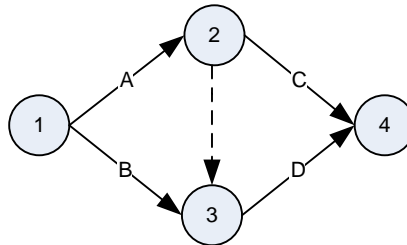


Figura 4 - actividades nos arcos com actividades fictícias

2. Como só há um arco entre dois nós, as actividades fictícias são necessárias para representar actividades paralelas

A lista de actividades, na qual não há precedências: A; B; C poderá ter a representação da Figura 5.

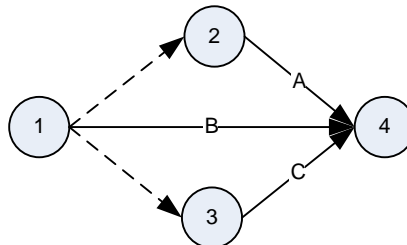


Figura 5 - actividades nos arcos com actividades fictícias para representar actividades paralelas

3. Como só pode existir um nó de início e um nó de fim.

A lista de precedências: A; B; C(A,B); D(A,B) poderá ter a representação da Figura 6.

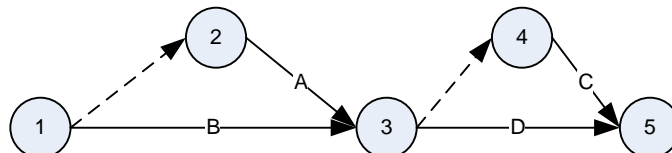


Figura 6 - actividades nos arcos com actividades fictícias para só ter um nó de início e um nó final

1.3.5.3 Número mínimo e máximo de arcos

Dado o número de nós N , o número mínimo de arcos acontece quando todas as actividades são executadas em sequência e é igual a $N-1$.

Dado o número de nós N , o número máximo de arcos acontece quando todos os nós têm um arco para os nós que lhe sucedem e é igual a $N*(N-1)/2$ (ver Equação 1).

$$A = (N - 1) + (N - 2) + \dots + 1 = \sum_{i=1}^{N-1} i = \frac{N * (N - 1)}{2}$$

Equação 1 - calculo do número máximo de arcos

1.3.5.4 Matrizes de adjacência para representar grafos

Quando se pretende verificar a existência de um arco entre dois nós basta verificar a representação gráfica, mas essa representação tem pouca utilidade quando se pretende efectuar operações computacionais sobre o grafo. A matriz de adjacências é uma representação mais correcta para computador.

Dado o número de nós N , a matriz de adjacências, de um grafo, é uma matriz quadrada de dimensão N ,

$$A = [a_{ij}] \quad 1 \leq i, j \leq \text{número de nós}$$

tal que $a_{ij}=1$ se existir arco de i para j e $a_{ij}=0$ se não existir arcos de i para j .

No caso na representação de projectos, a matriz de adjacências tem posições nas quais o valor é sempre zero (representados com fundo mais escuro na Figura 7) e pode ocupar menos memória no computador se for utilizada uma matriz triangular superior.

Nota: no resto do documento, para facilitar a leitura, não serão representados os valores constantes.

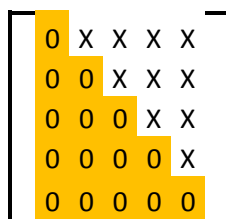


Figura 7 – matriz triangular superior de uma rede com 5 nós

1.3.6 Engine Tester

O *Engine Tester* é um software desenvolvido pelo Professor Doutor José Coelho, Universidade Aberta com objectivo de efectuar testes de software.

As principais funcionalidades do *Engine Tester*[6] são os testes empíricos em algoritmos e os testes de componentes.

Os testes de componentes tem como objectivo o teste de classes, módulos ou componentes após estes estarem completos e podem ser efectuados pelo programador mas também devem ser efectuados outra pessoa.

Os testes empíricos consistem na execução do código, para verificar se os resultados obtidos estão correctos tendo em conta os argumentos.

O *Engine Tester* permite o encadeamento de algoritmos da forma que a instância de saída de um determinado algoritmo seja a instância de entrada de outro algoritmo.

Permite também que, os algoritmos sejam implementados em qualquer linguagem de programação, desde que em “*Console Application*” e seja implantada a interface de comunicação que está definida. Com o programa é disponibilizada, na linguagem de programação C++, a interface de comunicação.

2 Gerador aleatório de actividades

2.1 Principais características do gerador

O gerador criado é baseado no artigo “*A Random Activity Network Generator*”[1] e tem como principais características:

- São geradas instâncias aleatórias de problemas do tipo “*Resource Constrained Project Scheduling Problem*” RCPSP;
- A representação interna das precedências é do tipo *actividades nos arcos* sem utilização de actividades fictícias;
- As precedências são todas do tipo *fim início*, sem tempo de espera;
- Os recursos são todos renováveis;
- É permitido o controlo dos seguintes parâmetros do gerador:
 - Rede:
 - Número de arcos;
 - Número de nós;
 - Recurso do projecto:
 - Número de recursos disponíveis;
 - Disponibilidade de cada recurso;
 - Actividade:
 - Duração;
 - Número de recursos necessários;
 - Ocupação de cada recurso;
- A instância de saída tem o formato *PSPLib* (ver Apêndices II – Exemplo instância saída).

2.2 Geração da rede

Dado o número de arcos e de nós a rede de precedências é escolhida aleatoriamente de todas as redes possíveis tendo todas igual probabilidade.

Verifica-se na Figura 8 que a probabilidade do nó de origem vai diminuindo à medida que a índice do nó vai aumentando.

1	1	1	1	Probabilidade (origem=1)=4/10=2/5
	1	1	1	Probabilidade (origem=2)=3/10
		1	1	Probabilidade (origem=3)=2/10=1/5
			1	Probabilidade (origem=4)=1/10
				Probabilidade (origem=5)=0

Figura 8 - matriz de adjacências completa para uma rede com 5 nós

A geração da origem de um arco terá de atender ao facto dos nós de origem terem diferentes probabilidades. A Equação 2[1] entra em conta com essa diferença.

$$\text{nó origem} = \text{maior inteiro menor ou igual a} \\ (n^{\text{o}} \text{ arcos} - \sqrt{n^{\text{o}} \text{ arcos} * (n^{\text{o}} \text{ arcos} - 1) * U(0,1) * 0,25 + 0,5})$$

Equação 2³ - geração do nó origem do arco[1]

Todos os nós com índice superior ao nó de origem têm igual probabilidade de serem escolhidos. A Equação 3 apresenta a fórmula de escolha do nó de destino.

$$\text{nó destino} = \text{maior inteiro menor ou igual a} \\ (\text{nó origem} + 1 + U(0,1) * (n^{\text{o}} \text{ arcos} - \text{nó origem}))$$

Equação 3 - geração nó de destino do arco

Para gerar a rede de precedências pode ser utilizado um de dois métodos:

- Remoção;
- Adição.

2.2.1 Método de geração por remoção

Este método começa com arcos de todos os nós para os seus sucessores e depois remove-os, utilizando as Equações 2 e 3, até ficar só com o número de arcos necessários (ver Equação 4).

³ U(inferior , superior) – distribuição uniforme entre inferior e superior

$$n^{\circ} \text{ arcos remover} = n^{\circ} \text{ máximo de arcos} - n^{\circ} \text{ arcos gerar}$$

$$n^{\circ} \text{ arcos remover} = \frac{n^{\circ} \text{ nós} * (n^{\circ} \text{ nós} - 1)}{2} - n^{\circ} \text{ arcos gerar}$$

Equação 4 - número arcos remover a remover no método remoção

O método de geração por remoção de arcos tem que respeitar a alínea 6 do ponto 1.3.5.1 e pode ser descrito da seguinte forma:

- Todos os nós, à excepção do nó inicial, tem de ter pelo menos um arco a incidir.
- Todos os nós, à excepção do nó final, tem que ter pelo menos um arco a emergir.

2.2.2 Método de geração por adição

Este método começa com dois arcos, arco do primeiro para o segundo nó e do penúltimo para o último nó. Estes arcos têm de existir obrigatoriamente.

São gerados arcos livres, isto é, arcos que não são necessários para garantir que pelo menos um arco a sair e um arco a entrar nos nós, utilizando as Equações 2 e 3. Este processo perdura enquanto o resultado da Equação 5 for superior a zero.

$$\text{Arcos Livres} =$$

$$n^{\circ} \text{ arcos a gerar} - n^{\circ} \text{ arcos gerados} - n^{\circ} \text{ nós com arcos sair} - n^{\circ} \text{ nós com arcos a entrar}$$

Equação 5 - número arcos livres

Os nós que não tenham arcos a entrar são escolhidos como nós de destino. Todos os nós com índice inferior ao nó de destino têm igual probabilidade de serem escolhidos. A Equação 6 apresenta a forma de geração do nó de origem.

$$\text{nó origem} = \text{maior inteiro menor ou igual a}$$

$$((\text{nó destino} - 1) * U(0,1) + 1)$$

Equação 6 - geração nó de origem do arco

Os nós que não tenham arcos a sair são escolhidos como nó de origem. Todos os nós com índice superior ao nó de origem têm igual probabilidade de serem escolhidos. A Equação 3 apresenta a forma de geração do nó de destino.

Como podem ser adicionados mais arcos que os necessários, os arcos em excesso serão removidos pelo método de remoção. O número de arcos a remover é apresentado na Equação 7.

$$\text{número arcos a remover} = \text{número arcos gerados} - \text{número de arcos a gerar}$$

Equação 7 - número de arcos a remover após aplicação método de adição

2.2.3 Escolha do método de geração

A escolha do método de geração é efectuada automaticamente pelo algoritmo do programa implementado, tendo em atenção o menor número de operações, de adição ou remoção de arcos, necessárias. O valor fronteira de escolha entre os métodos é quando ambos efectuem o mesmo número de operações (ver Equação 8). Quando o número de arcos a gerar for menor que o valor fronteira utiliza-se o método de adição caso contrário utiliza-se o método de remoção. Foi definido que quando for igual, utilizar-se o método de remoção porque não implicar a utilização do outro método.

$$\begin{aligned} n^{\circ} \text{ arcos gerar} &= \frac{n^{\circ} \text{ nós} * (n^{\circ} \text{ nós} - 1)}{2} - n^{\circ} \text{ arcos gerar} \\ \text{valor fronteira} &= \frac{n^{\circ} \text{ nós} * (n^{\circ} \text{ nós} - 1)}{4} \end{aligned}$$

Equação 8 - número de arcos a gerar em que a quantidade de operações é igual em ambos os métodos

2.3 Geração de parâmetros

Como existe uma infinidade de valores possíveis para os parâmetros não é possível garantir que qualquer valor tenha igual probabilidade.

Cada parâmetro pode ser:

- Gerado automaticamente (por omissão);
- Atribuído através da definição de um valor fixo;
- Gerado de uma das seguintes distribuições:
 - Uniforme;
 - Exponencial;
 - Gama;
 - Beta;
 - Normal;
 - Poisson;
 - Binomial.

Para a atribuição de valor fixo ou a utilização de distribuições é necessário passar parâmetros.

2.3.1 Geração automática de valores

Os vários valores foram escolhidos, após análise sumária de 20 instâncias escolhidas aleatoriamente das 2040 instâncias do *PSPLib RCPSP*. As principais diferenças entre as instâncias geradas e as instâncias apresentadas no *PSPLib RCPSP*, sendo o gerador mais abrangente, são:

- Número de actividades:
 - *PSPLib*: 30, 60, 90 e 120 actividades.
 - Gerador: qualquer valor entre 19 e 1770.
- Número de recursos disponíveis no projecto:
 - *PSPLib*: 4 recursos.
 - Gerador: entre 4 e 10.

Os vários parâmetros são gerados de uma distribuição uniforme.

- O número de nós foi definido com sendo um inteiro entre 20 e 60.
- O número de arcos foi definido com sendo um inteiro entre $(\text{número de nós} - 1)$ e $(\text{número de nós}) * (\text{número de nós} - 1) / 2$.
- O número de recursos disponíveis no projecto foi definido com sendo um inteiro entre 4 e 10.
- A disponibilidade de recurso foi definido com sendo um inteiro entre 20 e 50.
- A duração das actividades foi definido com sendo um inteiro entre 1 e 10.
- O número de recursos utilizados nas actividades foi definido com sendo um inteiro entre 1 e número de recursos disponíveis no projecto.
- A ocupação de um recurso numa actividade foi definido com sendo um inteiro entre 1 e 10.

3 Problemas detectados

Foram detectados alguns problemas no artigo “A *Random Activity Network Generator*”[1]. Neste capítulo pretende-se descrever os problemas detectados e as soluções que foram implementadas.

3.1 Falta de arcos livres na geração de redes por remoção

Quando as redes a gerar são pouco densas, ratio *número de arcos/número nós* baixo, pode acontecer que a rede fique sem arcos livres. Quando a rede não tem arcos livres o método de remoção fica bloqueado, não sendo possível acabar a rede.

A matriz representada na Figura 9 não permite representar uma rede com 7 nós e 7 arcos porque todos os arcos representados são necessários para que os nós tenham pelo menos um arco a emergir ou para que os nós tenham pelo menos um arco a incidir.

O artigo “A *Random Activity Network Generator*”[1] não apresenta solução para este problema.

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ & 0 & 1 & 1 & 0 & 0 \\ & & 0 & 0 & 1 & 0 \\ & & & 0 & 1 & 0 \\ & & & & 1 & 0 \\ & & & & & 1 \end{bmatrix}$$

Figura 9 - matriz demonstração do problema do método de remoção

Para resolver este problema, sempre que se verifica que a rede não tem arcos livres é adicionado um arco á rede.

A verificação da não existência de arcos livres, no pior caso (quando não há arcos livres), tem complexidade assintótica $O((\text{número de nós})^2)$ e por este motivo, só é efectuada esta verificação após existirem quatro ciclos sem haver operação de adição ou remoção de arcos.

Foram geradas mais de 1.000.000 de operações de adição / remoção de arcos, (Figura 10) e verificou-se que 4 ciclos contêm aproximadamente 100% das instâncias.

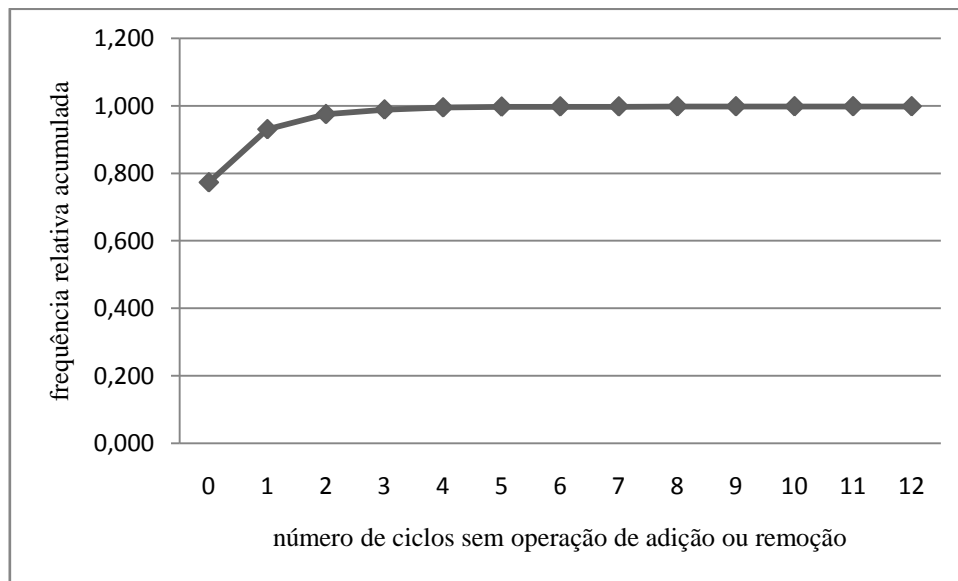


Figura 10- frequência relativa acumulada do número de ciclos sem operação de adição ou remoção

3.2 Não são gerados todos os projectos possíveis

Para representar todas as listas de precedências são necessárias actividades fictícias (ver ponto 1.3.5.2). Como não são geradas actividades fictícias (ver ponto 2.1), não é possível representar todas as listas de precedências.

Não foi proposta nenhuma solução a este problema porque a introdução de actividades fictícias é uma tarefa que deverá ser efectuada com cuidado, como se pode ser verificado pelos exemplos seguintes:

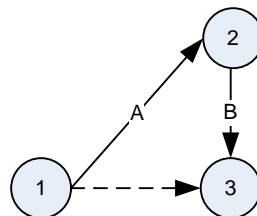


Figura 11 - problema na adição de actividades fictícias 1

Na rede representada na Figura 11 a introdução da actividade fictícia não tem qualquer mais valia visto que qualquer actividade que tenha de emergir do nó 3 tem sempre que esperar pelo termino da actividade B.

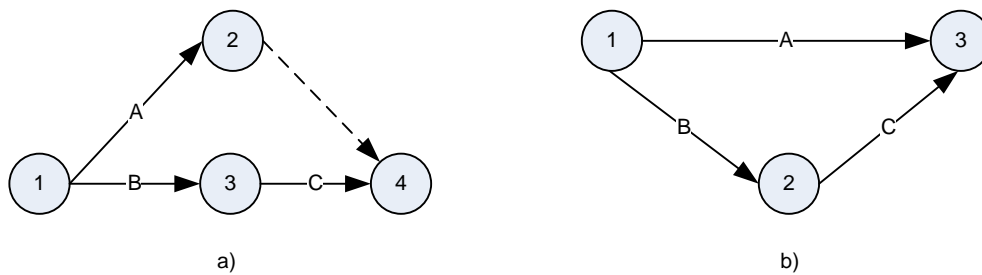


Figura 12 - problema na adição de actividades fictícias 2

Na Figura 12 duas representações distintas da mesma rede. A actividade fictícia não gerou uma nova rede.

3.3 Geração de lista de precedências com maior probabilidade que outras

Apesar de todas as matrizes terem igual probabilidade há listas de precedências com maior probabilidade.

Foram analisadas as 11 matrizes com 5 nós e 5 arcos. Na Figura 13 encontram-se todas as matrizes para essas redes.

$$\begin{array}{cccc}
 \begin{bmatrix} 1 & 1 & 0 & 0 \\ & 1 & 0 & 0 \\ & & 1 & 0 \\ & & & 1 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 0 & 0 \\ & 0 & 1 & 0 \\ & & 1 & 0 \\ & & & 1 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 0 & 0 \\ & 0 & 0 & 1 \\ & & 1 & 0 \\ & & & 1 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 0 & 0 \\ & 0 & 1 & 0 \\ & & 0 & 1 \\ & & & 1 \end{bmatrix} \\
 \text{a)} & \text{b)} & \text{c)} & \text{d)} \\
 \begin{bmatrix} 1 & 0 & 1 & 0 \\ & 1 & 0 & 0 \\ & & 1 & 0 \\ & & & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 1 & 0 \\ & 1 & 0 & 0 \\ & & 0 & 1 \\ & & & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 1 \\ & 1 & 0 & 0 \\ & & 1 & 0 \\ & & & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ & 1 & 1 & 0 \\ & & 1 & 0 \\ & & & 1 \end{bmatrix} \\
 \text{e)} & \text{f)} & \text{g)} & \text{h)} \\
 \begin{bmatrix} 1 & 0 & 0 & 0 \\ & 1 & 1 & 0 \\ & & 0 & 1 \\ & & & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ & 1 & 0 & 1 \\ & & 1 & 0 \\ & & & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ & 1 & 0 & 0 \\ & & 1 & 1 \\ & & & 1 \end{bmatrix} & \\
 \text{i)} & \text{j)} & \text{k)} &
 \end{array}$$

Figura 13 - todas matrizes de uma rede 5 nós e 5 arcos

A listas de precedências das matrizes são as seguintes:

- a) A;B;C(A);D(B,C);E(D)
- b) A;B;C(A);D(B);E(C,D)
- c) A;B;C(A);D(B);E(D)
- d) A;B;C(A);D(B);E(C)
- e) A;B;C(A);D(C);E(B,D)
- f) A;B;C(A);D(C);E(B)
- g) A;B;C(A);D(C);E(D)
- h) A;B(A);C(A);D(B);E(C,D)
- i) A;B(A);C(A);D(B);E(C)
- j) A;B(A);C(A);D(B);E(D)
- k) A;B(A);C(B);D(B);E(C)

Nota-se que listas de precedências c),d),f) são equivalentes. Na Figura 14 estão representadas as precedências através de diagrama de actividades nos nós. O resto das listas de precedências só tem uma representação, por esse facto, verifica-se que apesar

de todas as matrizes de adjacência terem igual probabilidade o mesmo não acontece no final com a lista de precedências.

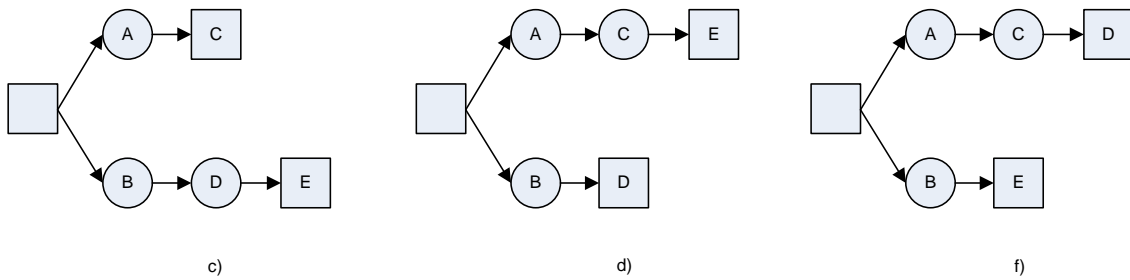


Figura 14 - diagrama de actividades nos nós equivalentes

Não é apresentada solução para este problema porque seria necessário conhecer de início todas as redes e respectivas probabilidades.

3.4 Geração do número de arcos.

Na Tabela 1 verifica-se que para número de nós superiores a 4 a distribuição de probabilidade do número de arcos não é completamente simétrica[1]. Tendo em conta este efeito no artigo “A Random Activity Network Generator”[1] é proposto utilizar-se uma distribuição normal com média apresentada na Equação 9 e desvio padrão apresentado na Equação 10.

Nós	Arcos														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1														
3		1	1												
4			1	4	4	1									
5				1	11	33	42	26	8	1					
6					1	26	171	507	840	865	584	262	76	13	1

Tabela 1 - número de redes possíveis dado o número de arcos e número de nós[1]

$$\mu = \frac{(L + U)}{2} - \frac{(U - L)^2}{500}$$

Equação 9 - média adaptada para geração do número de arcos[1]

$$\sigma = \frac{(U - L)}{2,5}$$

Equação 10 - desvio padrão para geração do número de arcos[1]

Na Figura 15 (Limite Inferior = número mínimo de arcos, Limite Superior = número máximo de arcos, Média = média entre Limite Inferior e Limite Superior e Média Adaptada = Equação 9) verifica-se que o valor da média adaptada fica inferior ao número mínimo de arcos quando o número de nó é superior a 24, por esse motivo esta forma de gerar número de arcos não se encontra implementada.

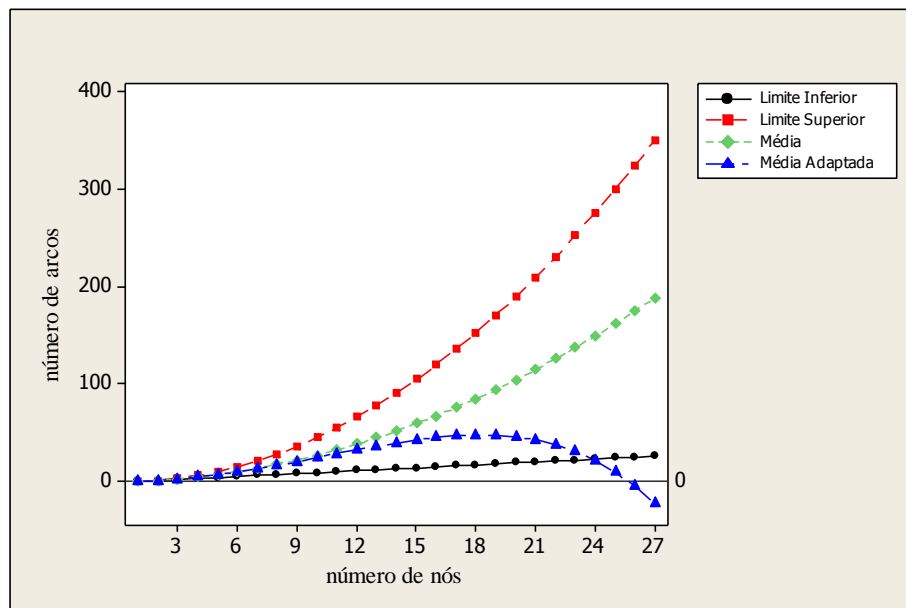


Figura 15 - número de arcos em função do número de nós

Método utilizado na geração do número de arcos é apresentado na Equação 11.

$$n^{\circ} \text{ de arcos} = U(n^{\circ} \text{ minimo de arcos}, n^{\circ} \text{ maximo de arcos})$$

Equação 11 - geração do número de arcos implementada

4 Implementação

O código foi desenvolvido em C++ e aproveitou-se a interface de comunicações que vem no pacote do *Engine Tester*.

4.1 Diagrama de classes

Na Figura 16 está representado o diagrama de classes das principais classes.

Por uma questão de legibilidade no diagrama não se encontram os atributos nem os operadores das classes, que poderão ser consultados no Apêndices III – Classes principais.

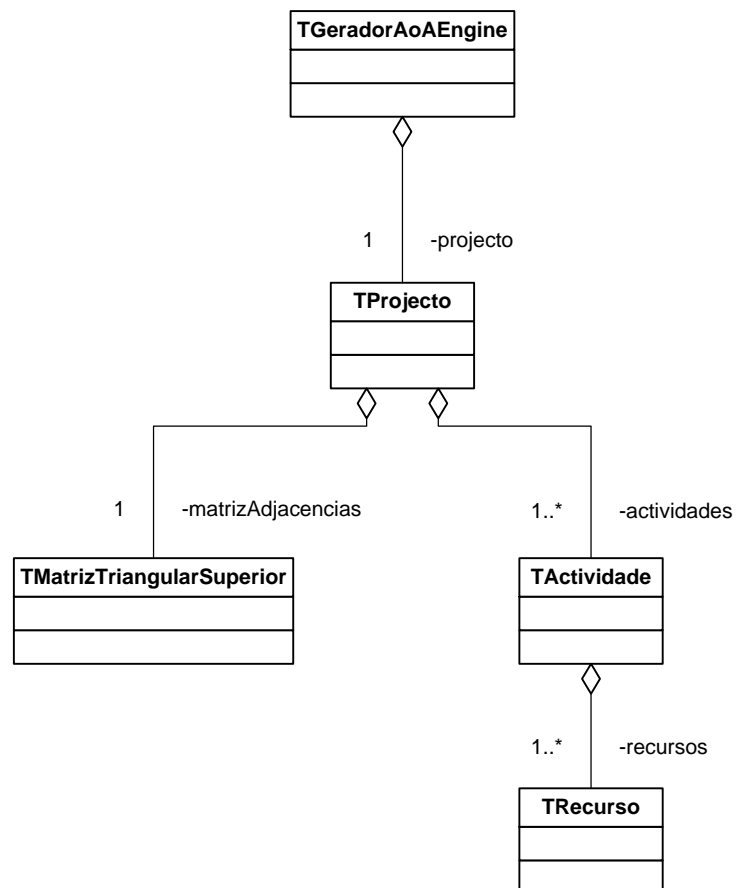


Figura 16 - diagrama das principais classes

4.2 Gerador Projecto

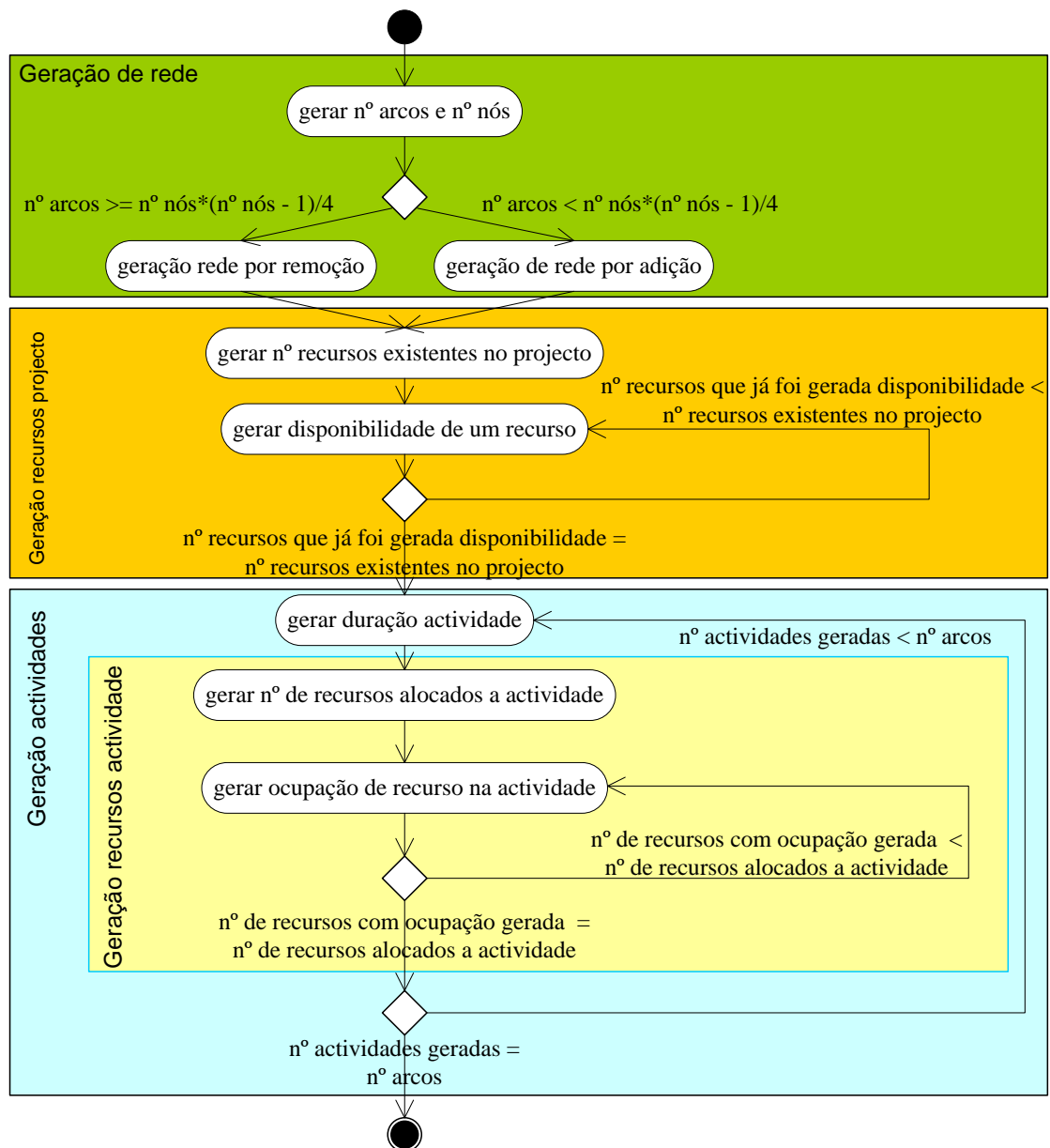


Figura 17 - geração projectos

A Figura 17 apresenta o diagrama de actividades do bloco central do programa, a geração de actividades nos arcos.

A geração de actividades nos arcos está dividida em 3 fases distintas, visíveis no diagrama por diferentes cores:

- Geração da rede.
- Geração recursos disponíveis no projecto

- Geração actividades.

4.3 Gerador de rede por remoção

São utilizadas duas estruturas auxiliares:

- in (nó) - número de arcos a incidir no nó;
- out (nó) - número de arcos a emergir no nó.

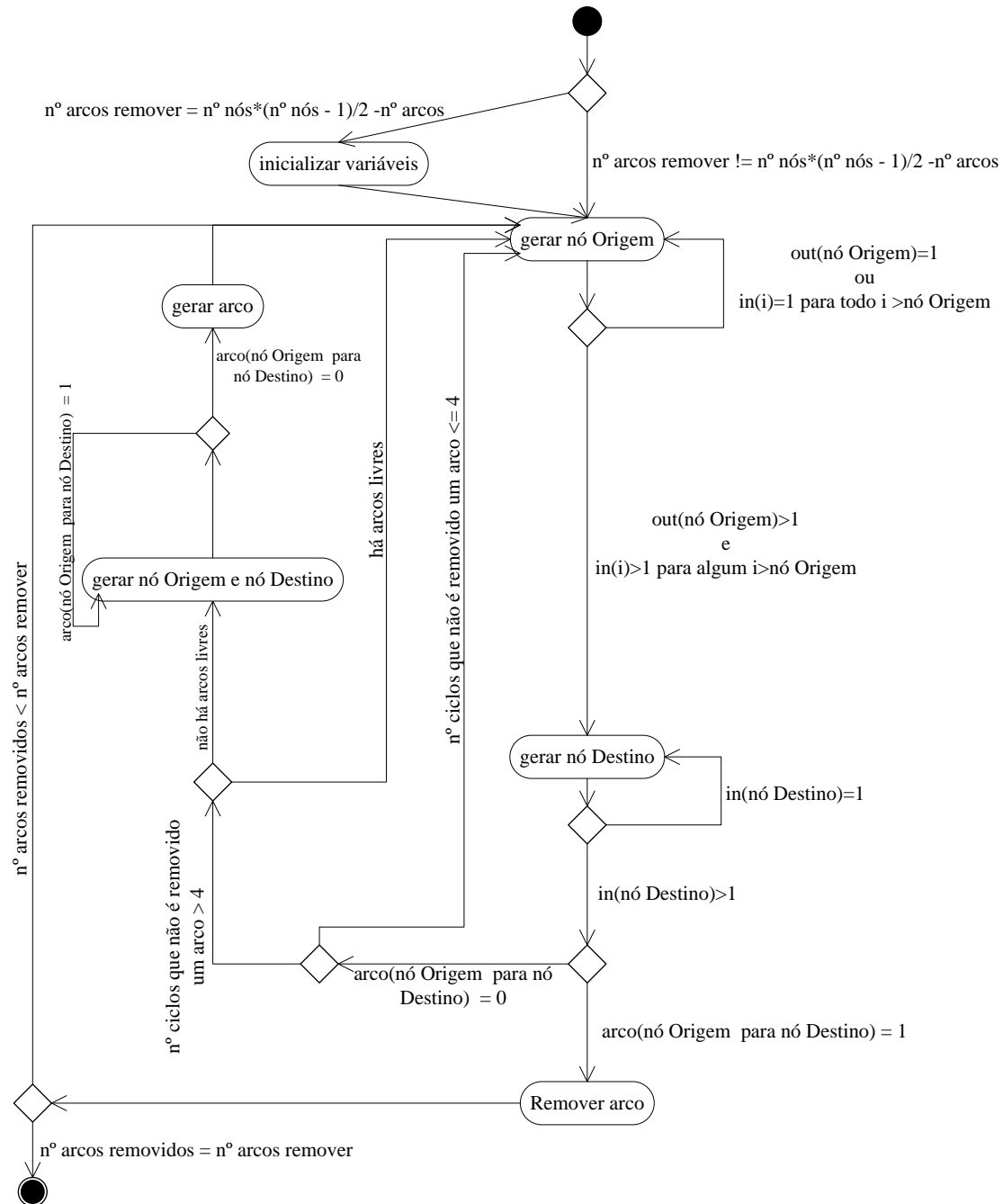


Figura 18 - geração de rede por remoção

A Figura 18 apresenta o diagrama de actividades da geração de rede por remoção.

Descrição das actividades da geração de rede por remoção:

- Inicializar variáveis
 - para toda origem = 1 até (nº nós - 1) e para todo destino = origem + 1 até nº nós fazer arco(origem para destino) = 1
 - para toda origem = 1 até (nº nós - 1) fazer out(origem) = nº nós - origem
 - para todo destino = 2 até (nº nós) fazer in(destino) = destino - 1
 - nº arcos removidos = 0
- Gerar nó Origem
 - origem = maior inteiro menor ou igual a $(\text{nº arcos} - \sqrt{\text{nº arcos} * (\text{nº arcos} - 1)} * U(0,1) * 0,25) + 0,5$
- Gerar nó Destino
 - destino = maior inteiro menor ou igual a $(\text{origem} + 1 + U(0,1) * (\text{nº arcos} - \text{origem}))$
- Remover arco
 - arco(origem para destino) = 0
 - out(origem) = out(origem) - 1
 - in(destino) = in(destino) - 1
 - nº arcos removidos = nº arcos removidos + 1
- Gerar nó Origem e nó Destino
 - origem = maior inteiro menor ou igual a $(\text{nº arcos} - \sqrt{\text{nº arcos} * (\text{nº arcos} - 1)} * U(0,1) * 0,25) + 0,5$
 - destino = maior inteiro menor ou igual a $(\text{origem} + 1 + U(0,1) * (\text{nº arcos} - \text{origem}))$
- Gerar arco
 - arco(origem para destino) = 1
 - out(origem) = out(origem) + 1
 - in(destino) = in(destino) + 1
 - nº arcos removidos = nº arcos removidos - 1

4.4 Gerador de rede por adição

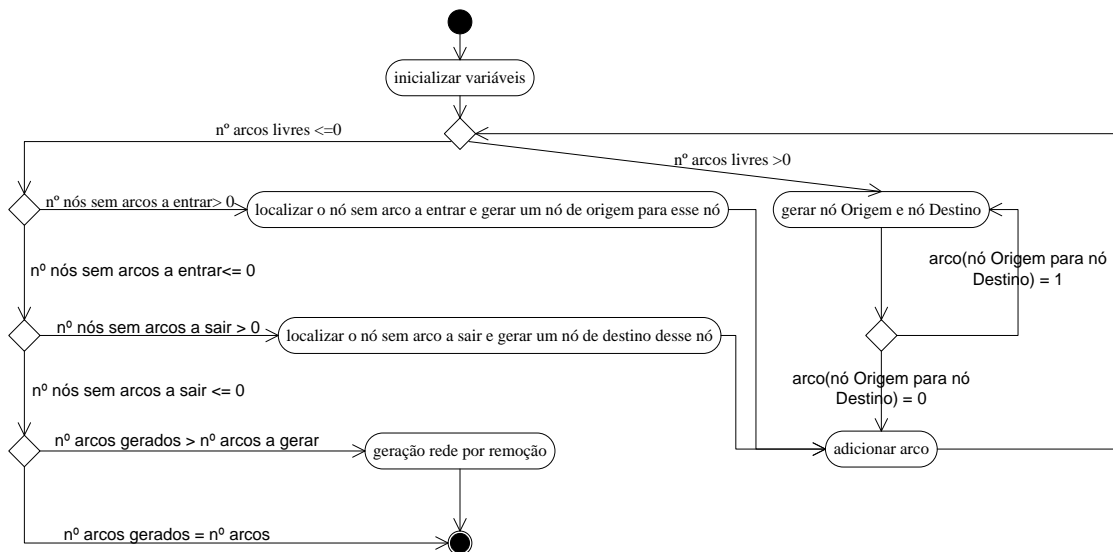


Figura 19 - geração rede por adição

A Figura 19 apresenta o diagrama de actividades da geração de rede por adição.

Descrição das actividades da geração de rede por adição:

- Inicializar variáveis
 - para toda origem = 2 até (nº nós - 1) e para todo destino = origem + 1 até nº nós arco fazer arco(origem para destino) = 0
 - arco(1 para 2) = 1
 - arco(nº nós - 1 para nº nós) = 1
 - para toda origem = 2 até (nº nós - 2) fazer out(origem) = 0
 - para todo destino = 3 até (nº nós - 1) fazer in(destino) = 0
 - out(1) = out(nº nós - 1) = in(2) = in(nº nós) = 1
 - nº arcos gerados = 2
 - nº nós sem arcos a sair = nº nós sem arcos a entrar = nº nós - 3
 - nº arcos livres = nº arcos a gerar - nº arcos gerados - nº nós sem arcos a sair - nº nós sem arcos a entrar
- Gerar nó Origem e nó Destino
 - origem = maior inteiro menor ou igual a $(n^{\circ} \text{ arcos} - \sqrt{n^{\circ} \text{ arcos} * (n^{\circ} \text{ arcos} - 1)} * U(0,1) * 0,25) + 0,5$

- destino = maior inteiro menor ou igual a $(origem+1+U(0,1)*(n^{\circ} \text{ arcos}-origem))$
- Actualiza valores
 - se $out(origem) == 0$ então $n^{\circ} \text{ nós sem arcos a sair} = n^{\circ} \text{ nós sem arcos a sair} - 1$
 - se $in(destino) == 0$ então $n^{\circ} \text{ nós sem arcos a entrar} = n^{\circ} \text{ nós sem arcos a entrar} - 1$
 - $arco(origem \text{ para } destino) = 1$
 - $out(origem) = out(origem) + 1$
 - $in(destino) = in(destino) + 1$
 - $n^{\circ} \text{ arcos gerados} = n^{\circ} \text{ arcos gerados} + 1$
 - $n^{\circ} \text{ arcos livres} = n^{\circ} \text{ arcos a gerar} - n^{\circ} \text{ arcos gerados} - n^{\circ} \text{ nós sem arcos a sair} - n^{\circ} \text{ nós sem arcos a entrar}$
- Localizar o nó sem arco a entrar e gerar um nó de origem para esse nó
 - destino = a um nó com $in(nó) == 0$
 - origem = maior inteiro menor ou igual a $((destino-1)*U(0,1)+1)$
- Localizar o nó sem arco a sair e gerar um nó de destino desse nó
 - origem = a um nó com $out(nó) == 0$
 - destino = maior inteiro menor ou igual a $(origem+1+U(0,1)*(n^{\circ} \text{ arcos}-origem))$
- Geração rede por remoção
 - Utilizar o método de remoção para remover $(n^{\circ} \text{ arcos gerados} - n^{\circ} \text{ arcos a gerar})$ arcos.

5 Testes

Os testes foram efectuados num Intel Pentium M a 1,6GHz com 1,25GB de RAM com Windows XP Professional.

Utilizou-se o Engine Tester na execução dos testes.

5.1 Testes de geração aleatória

Foram geradas 1.000 instâncias. Não foi passado nenhum parâmetro (geração automática) para além da semente utilizada na geração dos números aleatórios, que variou entre 100 e 999.100.

Na Figura 20 verifica-se que nas instâncias geradas os valores dos indicadores (número de nós, número de arcos e número de recursos) cobrem todo universo possível.

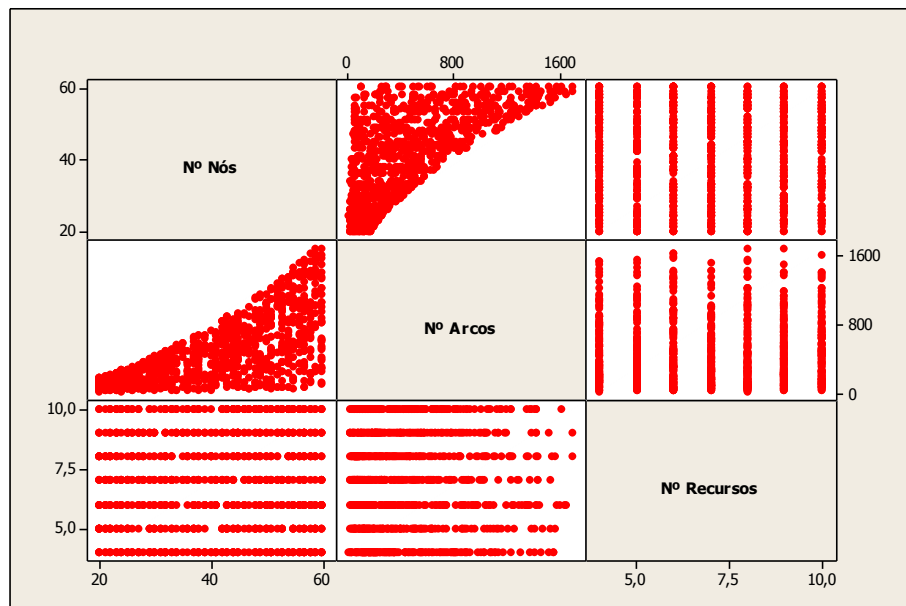


Figura 20 - variação de nº nós / nº arcos / nº recursos

Repara-se que nem todo o número de nós permite gerar todo número de arcos. Este efeito deve-se ao facto do número de arcos estar entre (número de nós - 1) e (número de nós * (número de nós - 1) / 2).

A existência de um maior número de arcos nos valores mais baixos é esperada porque qualquer número de nós contribui para estes arcos ao contrário do que acontece quando o número de arcos é mais elevado.

Na Figura 21 permite-nos verificar que o tempo de execução é quase sempre baixo. As excepções são quando a diferença entre o número de arcos - número de nós é muito baixo. Isto deve-se normalmente ao facto de quanto menor for o ratio *número de arcos/número nós* no método de remoção:

- Maior será o número de arcos que é necessário inserir para posterior remoção, devido à não existência de arcos livres (solução apresentada no ponto 3.1).
- Ou mesmo com arcos livres, serem necessários mais ciclos de procura sem remoção.

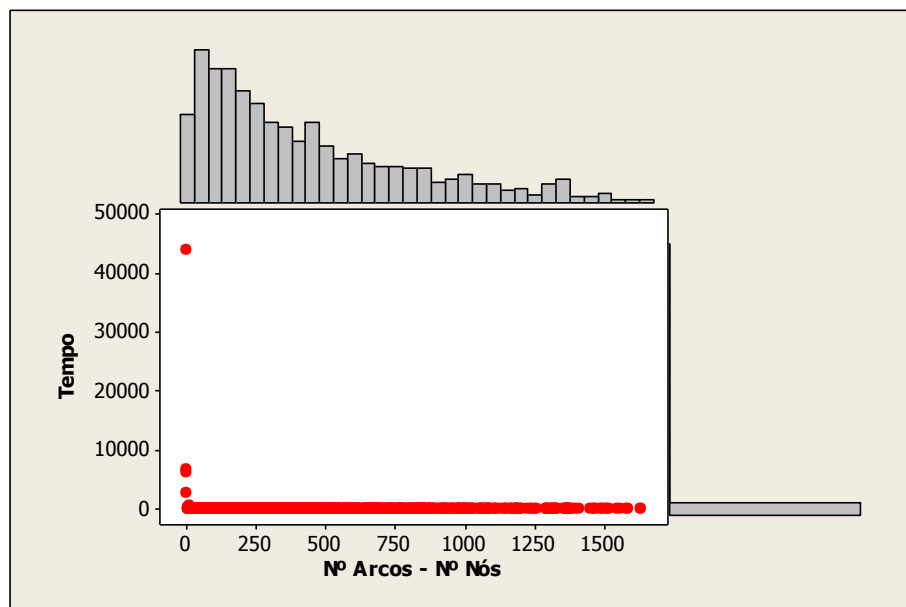


Figura 21 - tempo de execução do algoritmo pela diferença do nº arcos - nº nós

5.2 Testes de velocidade

Foram geradas 50 instâncias com 5.000 nós cada e variou-se do número de arcos de 20.000 até 1.000.000. Os outros parâmetros não foram passados (geração automática). Manteve-se a semente constante.

Na Figura 22 verifica-se uma relação linear entre a o tempo de execução e o número de arcos. Esta relação é confirmada na Figura 23.

Na mesma Figura 22 também verifica-se (linhas azul tracejado) que há uma relação entre o tempo de execução e a quantidade de recursos disponíveis no projecto. A Figura 24 vem confirmar a influência do número de recursos no tempo de execução.

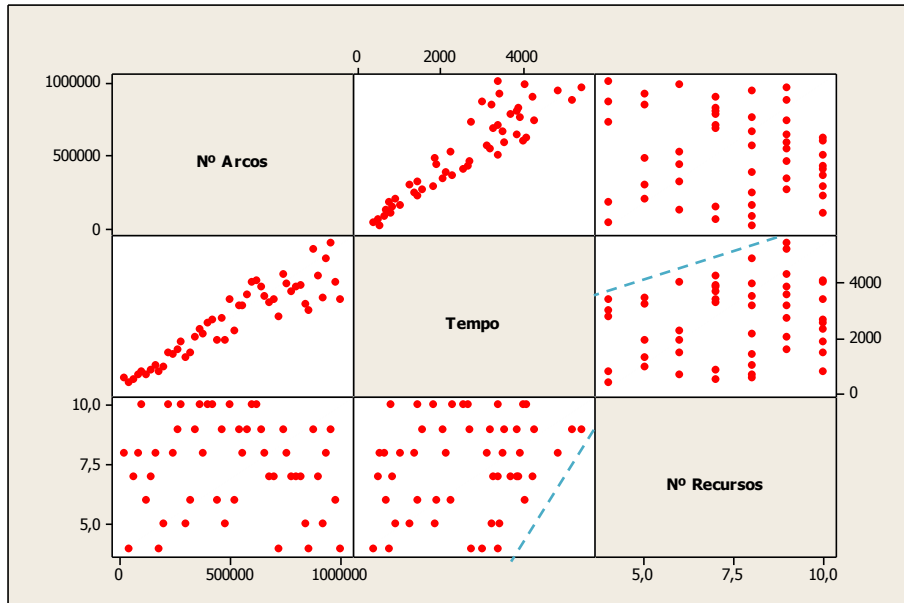


Figura 22 - variação de nº arcos / tempo / nº recursos

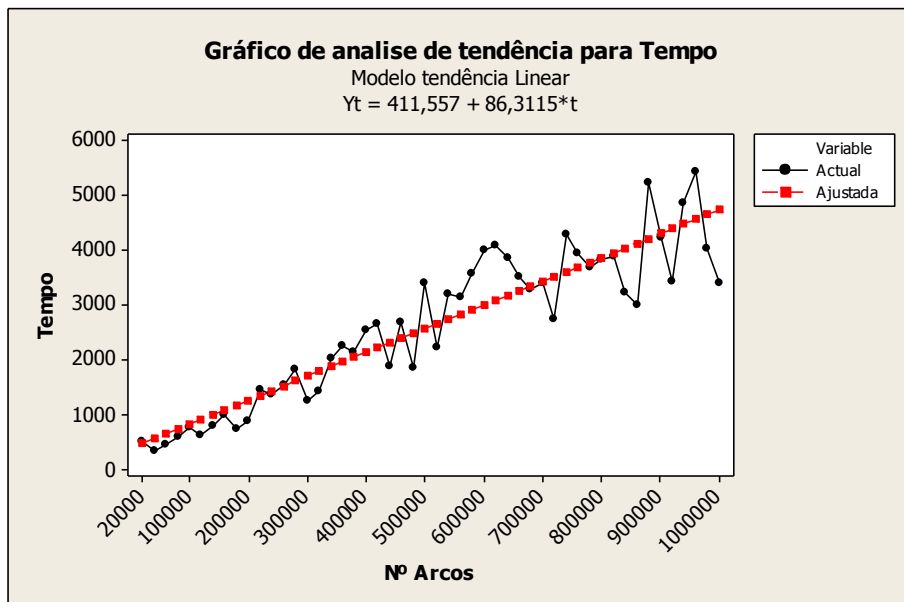


Figura 23 - gráfico de análise de tendência do tempo de execução

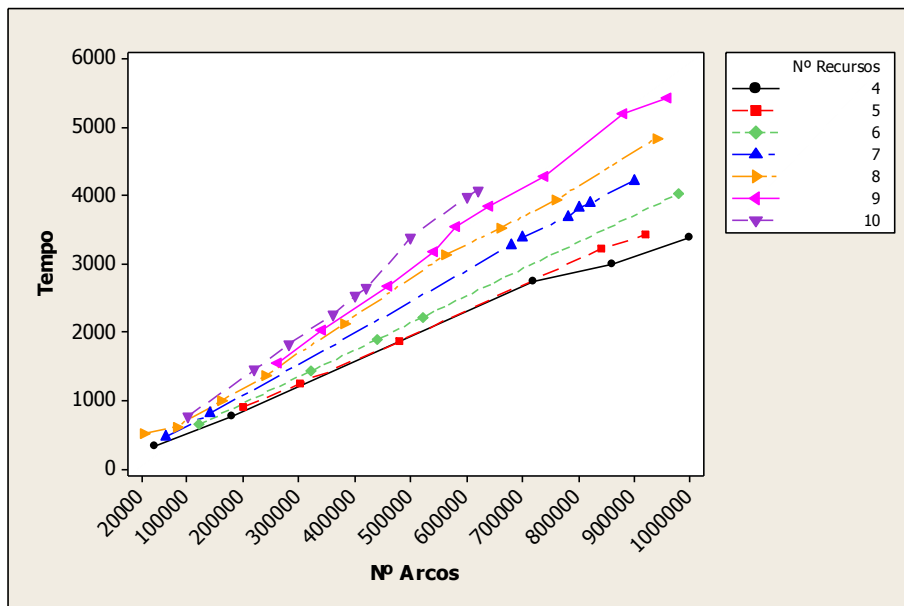


Figura 24 - tempo execução / nº arcos com grupos no nº de recursos

O tempo de execução passou os 5 segundos mas estes valores acontecem para uma quantidade elevada de arcos (960.000).

Para verificar a velocidade com quantidades inferiores de arcos foram geradas 80 instâncias com 250 nós e variou-se do número de arcos de 500 até 20.250. Os outros parâmetros não foram passados (geração automática). Manteve-se a semente constante. Na Figura 25 verifica-se que há algumas instâncias que têm tempos de execução muito baixos. Mesmo os valores mais elevados de tempo de execução não chegam a 150 milissegundos.

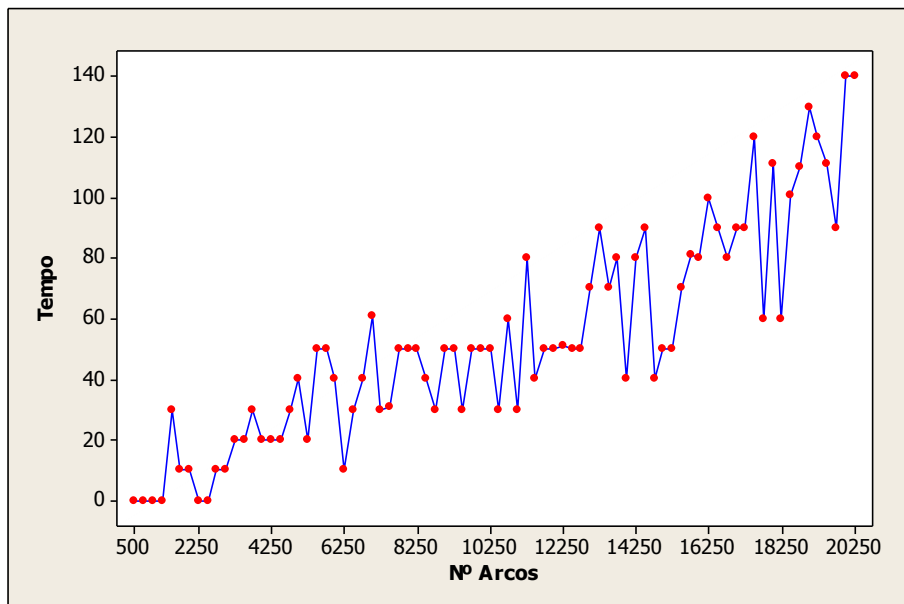


Figura 25 - tempo / nº arcos

5.3 Testes variação do número de nós mantendo o número de arcos

Foram geradas 685 instâncias com 1.000 arcos cada e variou-se do número de nós de 46 até 730. Os outros parâmetros não foram passados (geração automática). Manteve-se a semente constante.

Na Figura 26 verifica-se que os tempos de execução mantêm-se constante, e num valor baixo até os número de nós ser próximo dos 660.

Foi analisado o tempo de execução no qual o número de nós é superior a 667 e verificou-se que o tempo de execução cresce de forma exponencial (ver Figura 27). Este crescimento deve-se normalmente ao facto de quanto menor for o ratio *número de arcos/número nós* no método de remoção:

- Maior será o número de arcos que é necessário inserir para posterior remoção, devido à não existência de arcos livres (solução apresentada no ponto 3.1).
- Ou mesmo com arcos livres, serem necessários mais ciclos de procura sem remoção.

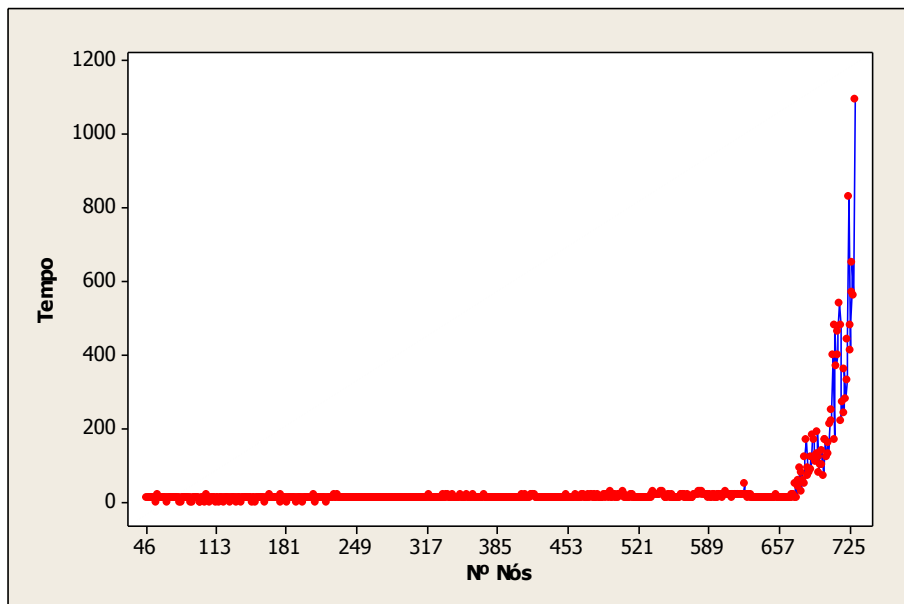


Figura 26 - tempo execução / nº nós

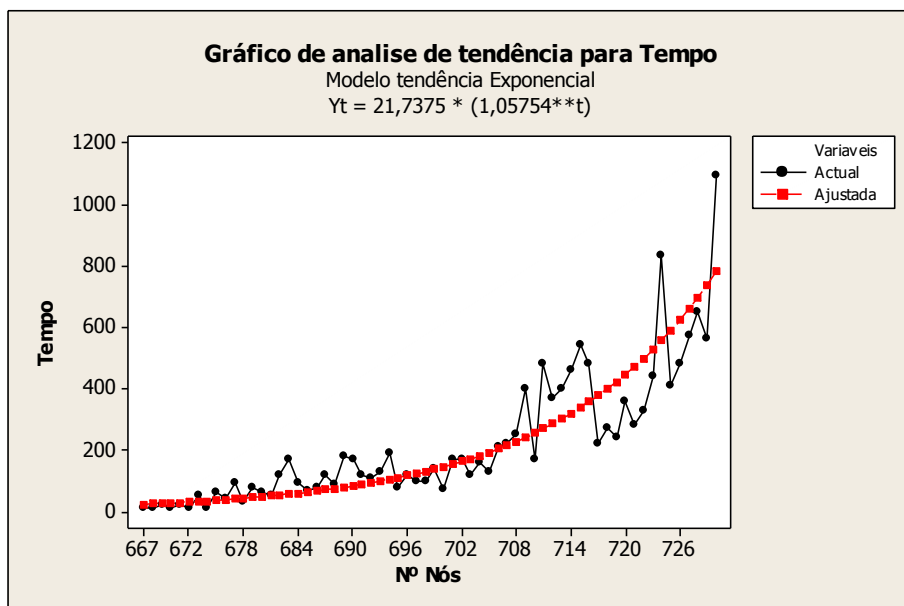


Figura 27 - gráfico de análise de tendência do tempo de execução

6 Conclusões

Neste trabalho foi implementado um gerador de instâncias aleatórias de problemas do tipo *RCPSP* que tem a possibilidade de ser utilizado como gerador de instâncias no *Engine Tester*.

O gerador, conjuntamente com o *Engine Tester*, são ferramentas que permitem submeter os métodos de resolução de problemas de calendarização a um conjunto grande de problemas aleatórios, de uma forma simples.

O gerador permite o controlo de alguns parâmetros de entrada, permitindo a geração de instâncias com alguma particularidade. Por exemplo, a duração das actividades pode ter uma distribuição normal.

Em termos de tempos de execução, desde que o ratio *número de arcos/número nós* seja superior a 1,5, o gerador é bastante eficiente sendo possível gerar-se projectos com 100.000 actividades em menos de 1 segundos.

Como é apresentado no capítulo de problemas, actualmente o gerador não cria todas instâncias possíveis de problemas do tipo *RCPSP*, devido a não trabalhar com actividades fictícias. Para trabalho futuro propõem-se o estudo de um método de introduzir actividades fictícias, sem impactos negativos.

Referências

- [1] E. Demeulemeester, B. Dodin, and W. Herroelen, "A Random Activity Network Generator," *Operations Research*, vol. 41, no. 5, pp. 972-980, 1993.
- [2] J. Coelho. "Engine Tester," 2008; <http://jcoelho.m6.net/edicao3.asp?pa=4229>.
- [3] A. Drex1, R. Nissen, J. H. Patterson *et al.*, "ProGen/px—An instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions," *European Journal of Operational Research*, vol. 125, no. 1, pp. 59-72, 2000.
- [4] J. Coelho. "Sebenta de Gestão de Projectos Informáticos 07/08," 2008; <http://www.moodle.univ-ab.pt/moodle/mod/book/index.php?id=564>.
- [5] E. L. Demeulemeester, *Project scheduling : a research handbook*, 2002.
- [6] J. Coelho. "Sebenta de Desenvolvimento de Software 07/08," 2008; <http://www.moodle.univ-ab.pt/moodle/mod/book/view.php?id=15451>.

Acrónimos

AoA	Actividades nos Arcos
AoN	Actividades nos Nós
PSPLIB	<i>Project Scheduling Problem Library</i>
RCPSP	<i>Resource Constrained Project Scheduling Problem</i>

Apêndices I - Manual de utilizador

O presente programa implementa o algoritmo apresentado no artigo “*A Random Activity Network Generator*”[1] que permite gerar redes actividades nos arcos, seguindo a filosofia do Engine Tester[2].

O gerador apresentado é fortemente aleatório, possibilitando o controlo da distribuição utilizada na geração dos seguintes parâmetros:

- Número arcos;
- Número nós;
- Duração das actividades;
- Número de recursos disponíveis no projecto;
- Quantidade de cada recurso disponível no projecto;
- Número recursos atribuídos a cada actividade;
- Ocupação de recurso em cada actividade.

Para cada um dos parâmetros anteriores pode ser gerada uma variável aleatória com uma das seguintes distribuições (o número de parâmetros depende do tipo de distribuição).

Tipo Distribuição	Parâmetros de tipo distribuição	Parâmetro 1	Parâmetro 2
Valor passado no parâmetro	0	valor utilizar	N/A ⁴
Uniforme	1	limite inferior	limite superior
Exponencial	2	Escala	N/A
Gamma	3	Forma	escala
Beta	4	1º parâmetro forma	2º parâmetro forma
Normal	5	Média	desvio padrão
Poisson	6	Média	N/A
Binomial	7	probabilidade sucesso	nº repetições
Geração automática de valores (por omissão ⁵)	8	N/A	N/A

⁴ N/A – Não aplicável

⁵ Se não for passado outro tipo distribuição, todos os parâmetros do projecto são gerados de distribuições pré-definidas

Por definição inicial, os vários parâmetros são gerados de uma distribuição uniforme.

- O número de nós foi definido com sendo um inteiro entre 20 e 60.
- O número de arcos foi definido com sendo um inteiro entre (número de nós – 1) e (número de nós)* (número de nós -1)/2.
- O número de recursos disponíveis no projecto foi definido com sendo um inteiro entre 4 e 10.
- A disponibilidade de recurso foi definido com sendo um inteiro entre 20 e 50.
- A duração das actividades foi definido com sendo um inteiro entre 1 e 10.
- O número de recursos utilizados nas actividades foi definido com sendo um inteiro entre 1 e número de recursos disponíveis no projecto.
- A ocupação de um recurso numa actividade foi definido com sendo um inteiro entre 1 e 10.

Parâmetros de entrada:

ID do parâmetro	Descrição do parâmetro
1	Número arcos - Tipo de distribuição
2	Número arcos - Parâmetro 1
3	Número arcos - Parâmetro 2
4	Número nós - Tipo de distribuição
5	Número nós - Parâmetro 1
6	Número nós - Parâmetro 2
7	Duração actividade - Tipo de distribuição
8	Duração actividade - Parâmetro 1
9	Duração actividade - Parâmetro 2
10	Número recursos disponíveis no projecto - Tipo de distribuição
11	Número recursos disponíveis no projecto - Parâmetro 1
12	Número recursos disponíveis no projecto - Parâmetro 2
13	Quantidade de cada recurso disponível no projecto - Tipo de distribuição
14	Quantidade de cada recurso disponível no projecto - Parâmetro 1
15	Quantidade de cada recurso disponível no projecto - Parâmetro 2
16	Número recursos atribuídos a cada actividade - Tipo de distribuição
17	Número recursos atribuídos a cada actividade - Parâmetro 1
18	Número recursos atribuídos a cada actividade - Parâmetro 2
19	Ocupação de recurso em cada actividade - Tipo de distribuição
20	Ocupação de recurso em cada actividade - Parâmetro 1
21	Ocupação de recurso em cada actividade - Parâmetro 2

Indicadores disponíveis:

ID do indicador	Descrição do indicador
1	Número de arcos da rede do projecto
2	Número de nós da rede do projecto
3	Número de recursos renováveis disponíveis no projecto

Comandos:

Além de poder ser utilizado conjuntamente com o Engine Tester o programa pode ser utilizado isoladamente utilizando os seguintes comando.

Comandos de corrida:

- RUN – inicia a geração do projecto
- STOP – força a paragem da geração
- RESET – reinicia todos os dados do gerador

Comandos de configuração:

- SET [k] [valor] – coloca um valor no parâmetro k
- SET SEED [valor] – coloca a semente do gerador de números aleatórios
- SET LIMIT TIME [milissegundos] – limita o tempo utilizado para a geração

Comandos de retirar informação:

- GET INSTANCE – devolve o caso actual
- GET FILE [nome ficheiro] – salva o caso actual para um ficheiro
- GET TIME – devolve o tempo utilizado na geração, em milissegundos
- GET SETTINGS – devolve o número de parâmetros disponíveis
- GET [k] – devolve o valor do parâmetro k (valor inteiro)
- GET NAME [k] – devolve o nome do parâmetro k
- GET DESCRIPTION [k] – devolve a descrição do parâmetro k
- GET MINIMAL [k] – devolve o valor mínimo do parâmetro k
- GET MAXIMAL [k] - devolve o valor máximo do parâmetro k
- GET INDICATORS – devolve o número de indicadores disponíveis
- GET INDICATOR [k] – devolve o valor do indicador k
- GET INDICATOR NAME [k] – devolve o nome do indicador k
- GET INDICATOR DESCRIPTION [k] – devolve a descrição do indicador k

- GET ENGINE NAME – devolve o nome do gerador
- GET ENGINE VERSION – devolve a versão do gerador
- GET ENGINE DATE – devolve a data de compilação do gerador
- GET ENGINE AUTHOR – devolve o nome dos autores do gerador
- GET ENGINE CONTACT – devolve informação de contacto sobre o gerador
- GET ENGINE DESCRIPTION – devolve uma pequena descrição do gerador
- GET LIMIT TIME – devolve o limite de tempo em milissegundos

Exemplos:

1. Gerar um caso completamente aleatório e gravar a instância de saída no ficheiro saída.sm

run

get file saída.sm

2. Gerar um caso com: semente igual a 100, 10 arcos, 5 nós, a duração da actividade é uma variável aleatório normal com média 10 e desvio padrão 2 e grava a instância de saída no ficheiro saída.sm.

set seed 100

set 1 0

set 2 10

set 4 0

set 5 5

set 7 5

set 8 10

set 9 2

run

get file saída.sm

Apêndices II – Exemplo instância saída

```

*****
file with basedata      :
initial value random generator: 100
*****

projects                :      1
jobs (incl. supersource/sink):    12
horizon                  :      0
RESOURCES
- renewable              :      7 R
- nonrenewable           :      0 N
- doubly constrained     :      0 D
*****

PROJECT INFORMATION:
prnrr. #jobs rel.date duedate tardcost MPM-Time
1      10      0      39      1      39
*****

PRECEDENCE RELATIONS:
jobnrr. #modes #successors successors
1      1      4      2 3 4 5
2      1      3      6 7 8
3      1      2      9 10
4      1      1      11
5      1      1      12
6      1      2      9 10
7      1      1      11
8      1      1      12
9      1      1      11
10     1      1      12
11     1      1      12
12     1      0
*****

REQUESTS/DURATIONS:
jobnrr. mode duration R 1 R 2 R 3 R 4 R 5 R 6 R 7
-----
1      1      0      0 0 0 0 0 0 0 0
2      1      7     10 0 7 0 9 0 9
3      1     12      7 4 0 0 0 8 9
4      1      7      0 2 4 7 6 10 5
5      1     11      4 5 2 8 10 9 9
6      1      9      0 6 0 5 7 1 4
7      1     11     10 10 3 0 0 7 7
8      1     12      8 0 0 0 0 0 0
9      1     13      5 4 2 0 10 9 4
10     1     10      5 0 9 0 1 1 2
11     1     10      7 1 5 4 7 7 5
12     1      0      0 0 0 0 0 0 0
*****

RESOURCEAVAILABILITIES:
R 1 R 2 R 3 R 4 R 5 R 6 R 7
40 26 34 24 27 28 24
*****

```

Apêndices III – Classes principais

Classe TGeradorAoAEngine

Uma das classes que vêm com o Engine Tester para implementar a interface de comunicação, esta é a classe que é alvo do maior número de modificações.

Operadores

```
<<construtor>> +TGeradorAoAEngine()  
<<destrutor>> +TGeradorAoAEngine()  
+Engine(type: int): char  
+Indicator(indicator: long): long  
+Load(f: FILE): void  
+LoadSolution(f: FILE): void  
+Reset(): void  
+Run(): void  
+Save(f: FILE): void  
+SaveSolution(f: FILE): void  
+Seed(value: long): void
```

Classe TProjecto

Classe central, que implementa um projecto e a geração do projecto aleatório.

Atributos

```
-in: TVector<long>  
-out: TVector<long>  
-primeiroIDSairNo: TVector<long>  
-recursos: TVector<int>  
-duracaoTotal: long  
-geracaoTerminada: bool  
-horizon: long  
-numeroAleatorio: RNG  
-numeroArcos: long  
-numeroNos: long  
-numeroRecursosNoProjecto: int  
-seed: unsigned long
```

Operadores

```
-DuracaoTotalCPM(): void  
-GeracaoRedePorRemocao(numeroArcosRemover: long): void  
-GeracaoRedePorAdicao(): void  
-NaoHaMaisArcosLivres(): bool  
-UpdateAM(noOrigem: long&, noDestino: long&, numeroNosSemArcosEntrar:  
long&, numeroNosSemArcosSair: long&, numeroArcosLivres: long&,  
numeroArcosGerados: long&): void  
<<construtor>> +TProjecto()  
<<destrutor>> +TProjecto(): void  
+Gerar(aleatorioNumeroNos: SParametroAletorios, aleatorioNumeroArcos:  
SParametroAletorios, aleatorioDuracaoActividade: SParametroAletorios,  
aleatorioNumeroRecursosAlocadosAoProjecto: SParametroAletorios,  
aleatorioDisponibilidadeRecursosNoProjecto: SParametroAletorios,  
aleatorioNumeroRecursosUtilizadosPorActividade: SParametroAletorios, aleatorioOcupacaoRecursos:  
SParametroAletorios): void  
+GetnumeroArcos(): long  
+GetNumeroNos(): long  
+GetNumeroRecursosNoProjecto(): long  
+Print(f: FILE): void  
+Seed(novaSemente: ulong): void
```

Classe TMatrizTriangularSuperior

Classe que implementa uma matriz triangular superior, de booleanos, necessária para representar a matriz de adjacências.

Atributos

-linhasCriadas: long
-linhasMaximas: long
-linhasUtilizacao: long
-matriz: bool

Operadores

<<construtor>> +TMatrizTriangularSuperior(: void)
<<destrutor>> +TMatrizTriangularSuperior(: void)
+Get(origem: long, destino: long): bool
+Init(nElementos: long, vInicial: bool): void
+Set(origem: long, destino: long, valor: bool): void

Classe TActividade

Classe que implementa uma actividade com determinada duração e recursos associados.

Atributos

duracao: int

Operadores

<<construtor>> +TActividade(: void)
<<destrutor>> +TActividade(: void)
+GetDuracao(): int
+GetNumeroRecursosAlocados(): int
+GetRecursoQuantidade(idRecurso: int): int
+Print(f: FILE, numeroRecursoProjecto: long): void
+SetDuracao(duracaoActividade: int): void
+SetRecurso(idRecurso: int, quantidadeRecurso: int): void

Classe TRecurso

Classe que implementa um recurso necessário para uma actividade.

Atributos

-id: int
-quantidade: int

Operadores

<<construtor>> +TRecurso(: void)
<<destrutor>> +TRecurso(: void)
+operator>(recurso: TRecurso): bool
+operator<(recurso: TRecurso): bool
+GetId(): int
+GetQuantidade(): int
+SetId(novoId: int): void
+SetQuantidade(novaQuantidade: int): void