

# Determinação de caminhos k-críticos em redes PERT

José Luiz Contador  
Edson Luiz França Senne



## Resumo

Neste trabalho, apresenta-se um estudo sobre os principais métodos para identificar os caminhos de maior duração em redes PERT, denominados na literatura de caminhos *k*-críticos (quando  $k = 1$ , tem-se o caminho mais longo, conhecido tradicionalmente por caminho crítico; quando  $k = 2$ , tem-se o segundo caminho mais longo, e assim sucessivamente). São discutidos três algoritmos apresentados na literatura e propõe-se um novo procedimento, denominado algoritmo da folga mínima, que apresenta algumas vantagens sobre os anteriores. O comportamento do algoritmo da folga mínima, quando aplicado a redes PERT, é verificado por meio de simulação.

**Palavras-chave:** Gerenciamento de projetos. PERT/CPM. Caminhos *k*-críticos.

## 1 Introdução

Um dos fatores decisivos para a competitividade em empresas de projeto é o fator tempo, que, assim como o fator custo, é considerado como um critério ganhador de pedidos. Assim, a redução do tempo de execução do projeto é um importante objetivo para essas empresas.

Para obter vantagem competitiva no tempo de execução do projeto, normalmente, seu planejamento é feito, primeiro, com base na duração esperada das suas atividades, gerando, assim, a duração esperada do projeto ( $T_e$ ), dada pela duração do seu caminho mais longo. Em seguida, estabelece-se uma meta,  $T^* < T_e$ , para sua realização e busca-se atingi-la acelerando os caminhos que possuem duração maior que  $T^*$ . Portanto, é fundamental identificar todos os caminhos que apresentam duração esperada superior a  $T^*$ .

Um caminho *k*-crítico numa rede refere-se ao caminho com a *k*-ésima maior duração. Quando  $k = 1$ , tem-se o caminho de maior duração, tradicionalmente conhecido por caminho crítico; quando  $k = 2$ , tem-se o caminho com a segunda maior duração, e assim sucessivamente. Interessa, portanto, conhecer os caminhos  $P_k$  do projeto,  $k=1, 2, \dots, n$ , cujas durações  $T(P_k)$  sejam maiores que  $T^*$ .

A literatura fornece três importantes algoritmos para determinação dos caminhos *k*-críticos em redes. Baseado

num procedimento de enumeração seletiva de caminhos, Dodin (1984) apresenta um método aproximativo para identificar os caminhos *k*-críticos mais longos em redes probabilísticas, isto é, redes em que a duração das atividades é uma variável aleatória. Yen et al. (1989) apresentam vários algoritmos para redes determinísticas, todos baseados num processo de ramificação e corte de caminhos parciais, o que acarreta baixa eficiência computacional, uma vez que muitos caminhos parciais inicialmente ramificados são abandonados ao longo do processo. Ju e Saleh (1991) apresentam um procedimento que evita esse inconveniente, o que o torna o algoritmo mais eficiente dentre os disponíveis na literatura. Esses três artigos serão analisados neste trabalho.

A literatura especializada não apresenta extensa lista de publicações para busca de métodos de determinação dos caminhos *k*-críticos em redes. Parece que o desenvolvimento de métodos se circunscreve principalmente a esses três artigos citados. Contudo, vários trabalhos mais recentes, que necessitam da identificação de caminhos *k*-críticos para seus propósitos, referenciam os artigos analisados neste texto. Esses trabalhos se concentram nas áreas de análise de projetos físicos e de análise de circuitos eletrônicos. Dentre aqueles pertencentes à

primeira destas áreas, pode-se relacionar Kim et al. (2007), Ghomi e Rabbani (2001) e Soroush (1994), que citam Dodin (1984). Soroush (1994) apresenta um método para determinar a probabilidade de cumprir prazo de conclusão do projeto cuja construção se assemelha ao processo utilizado por Dodin (1984). Ghomi e Rabbani (2001) apresentam um método para identificação da função de distribuição do tempo de conclusão de projetos em redes estocásticas que apresentou, em alguns casos, resultados melhores que aqueles obtidos por Soroush (1994). Kim et al. (2007) apresentam uma heurística para a alocação ótima de recursos em projetos para garantir, com alta probabilidade, a sua realização dentro de um prazo pré-definido. Dentre os trabalhos recentes relacionados com a área de eletrônica, pode-se citar Luo et al. (2006), referenciando Ju e Saleh (1991), e Borna et al. (2005), que referencia Yen et al. (1989). Ambos estudam o problema de atraso de resposta em circuitos eletrônicos e para isso é usual enumerar os caminhos que podem se tornar críticos.

Neste artigo, faz-se um estudo dos três métodos de busca de caminhos  $k$ -críticos citados e apresenta-se o algoritmo da folga mínima com o objetivo de identificar todos os caminhos numa rede PERT que possuem duração superior a um valor predefinido  $T^*$ , o qual se apresenta como uma alternativa ao algoritmo de Ju e Saleh (1991), pela maior facilidade de implementação. Esse novo algoritmo fundamenta-se em dois procedimentos: o primeiro, apresentado por Contador (2001), identifica, com grande eficiência, os caminhos da rede que estão explícitos pela folga das suas atividades; o segundo procedimento busca os eventuais caminhos que o primeiro não pôde identificar.

Este artigo está organizado da forma descrita a seguir. Na seção 2, são apresentadas a nomenclatura e as definições utilizadas ao longo do texto. Nas três seções seguintes, são apresentados os três algoritmos já disponíveis na literatura. Na seção 6, apresenta-se o algoritmo da folga mínima; na seção 7, são discutidos os ensaios computacionais realizados para avaliar a eficácia do primeiro procedimento embutido nesse algoritmo; e, na última seção, são apresentados os resultados e as conclusões do trabalho.

## 2 Nomenclatura e definições

A seguir são fornecidas a nomenclatura e as definições utilizadas ao longo do texto. Para mais detalhes sobre conceitos em redes PERT, pode-se consultar Moder et al. (1983).

Seja  $G(V, A)$  a rede representativa de um projeto em que  $A = \{a_r, r = 1, 2, \dots, m\}$  é o conjunto de arcos orientados da rede, representando as atividades do projeto, e  $V = \{v_i, i = 1, 2, \dots, n\}$  o conjunto de vértices, ou de nós, da rede.

Considere que aos vértices da rede existem atribuídos números  $i = 1, 2, \dots, n$  e que uma atividade é representada pelo par  $(i, j)$  correspondendo, respectivamente, aos seus vértices de início e de término.

A rede estará na sua forma canônica se todas suas atividades iniciais emergirem de um único vértice  $s$ , ao qual será atribuído o número 1, e todas as atividades finais incidirem também num único vértice  $t$ , ao qual será atribuído o número  $n$ . Para colocar a rede na forma canônica, criam-se os vértices  $s = 1$  e  $t = n$  e unem-se, por meio de arcos fictícios, o vértice  $s$  a cada um dos vértices iniciais da rede original e cada um dos vértices finais da rede original ao vértice  $t$ .

O vértice  $j$  é dito sucessor imediato de  $i$  se existir um e um só arco entre ambos, emergindo de  $i$  e incidindo em  $j$ . Nessas condições, o vértice  $i$  é chamado antecessor imediato de  $j$ .

Duas atividades são ditas atividades adjacentes se ambas têm o mesmo nó  $i$  como vértice de início (como as atividades  $(i, j)$  e  $(i, k)$ ), ou se ambas têm o mesmo nó  $k$  como vértice de término (como as atividades  $(i, k)$  e  $(j, k)$ ). Se duas atividades  $(i, j)$  e  $(j, k)$  têm o nó  $j$  como evento de término da primeira e evento de início da segunda, então a segunda atividade é dita sucessora imediata da primeira e esta, antecessora imediata da segunda.

A cada atividade  $(i, j) \in A$  serão associados os seguintes parâmetros:

- $T(i, j)$ , a variável aleatória duração da atividade  $(i, j)$ ,  $T(i, j) \in [to(i, j); tp(i, j)]$ ;
- $to(i, j)$ , a duração otimista da atividade  $(i, j)$ . Se  $(i, j)$  é fictícia, então  $to(i, j) = 0$ ;
- $tp(i, j)$ , a duração pessimista da atividade  $(i, j)$ . Se  $(i, j)$  é fictícia, então  $tp(i, j) = 0$ ;
- $tmp(i, j)$ , a duração mais provável da atividade  $(i, j)$ ;
- $t(i, j)$ , uma realização da variável  $T(i, j)$ ;
- $\bar{t}(i, j)$ , a média da variável  $T(i, j)$ ; e
- $s^2(i, j)$ , a variância da variável  $T(i, j)$ .

A média  $\bar{t}(i, j)$  e a variância  $s^2(i, j)$  da duração de uma atividade  $(i, j)$  podem ser estimadas por  $\bar{t}(i, j) = [to(i, j) + 4tmp(i, j) + tp(i, j)]/6$  e  $s^2(i, j) = [to(i, j) - tp(i, j)]^2/36$ .

Considerando-se a rede do projeto na forma canônica, dada uma realização  $t(i, j) \in T(i, j)$ , para toda atividade  $(i, j) \in A$ , define-se:

- data mais cedo de um vértice  $i$ , denotada por  $E(i)$  e determinada por  $E(i) = 0$  para  $i = 1$ , e  $E(i) = \max_{u_i} (E(u_i) + t(u_i, i))$ , para  $j = 2, 3, \dots, n$ , em que  $u_i$  são os vértices antecessores imediatos de  $i$ , como sendo o primeiro instante no qual todas as atividades que emergem de  $i$  podem ser iniciadas, ( $E(i)$  é igual à duração do caminho mais longo entre os vértices  $s$  e  $i$ );

- b) data mais tarde de um vértice  $i$ , denotado por  $L(i)$  e determinada por  $L(i) = E(i)$ , para  $i = n$  e  $L(i) = \min_{v_i} (L(v_i) - t(v_i, i))$ , para,  $i = n - 1, n - 2, \dots, 1$ , em que  $v_i$  são os vértices sucessores imediatos de  $i$ , como sendo o último instante no qual todas as atividades que incidem em  $i$  devem ser concluídas para que o projeto seja concluído na data  $L(n)$  ( $[L(n) - L(p)]$  é igual à duração do caminho mais longo entre o evento  $p$  e o evento  $t$ , fim do projeto); e
- c) folga da atividade  $(i, j)$ , denotada por  $f(i, j)$  e determinada por  $f(i, j) = L(j) - E(i) - t(i, j)$ , para todo  $(i, j) \in A$ .

Um caminho  $P$  na rede é um conjunto de arcos  $\{a_1, a_2, \dots, a_p\}$ , com  $a_i \in A$ , ( $i = 1, \dots, p$ ), tal que  $a_1 = (s, j)$  e  $a_p = (k, t)$ , sendo  $a_{i+1}$  sucessor imediato de  $a_i$ ,  $i = 1, \dots, p - 1$ . Obedecendo-se à orientação dos arcos da rede, por meio de um caminho é possível percorrer a rede e atingir o vértice  $t$  a partir do vértice  $s$ . Um caminho  $P$  pode ser representado também pelo conjunto de vértices que o compõe, ou seja,  $P = \{s, j, \dots, k, t\}$ .

Um caminho parcial  $P(i/j)$  na rede é um conjunto de arcos  $\{a_1, a_2, \dots, a_r\}$ , com  $a_i \in A$  ( $i = 1, \dots, r$ ), tal que  $a_1 = (i, k)$  e  $a_r = (q, j)$ , sendo  $a_{i+1}$  sucessor imediato de  $a_i$ ,  $i = 1, \dots, r - 1$ . Obedecendo-se à orientação dos arcos da rede, por meio de um caminho parcial  $P(i/j)$ , é possível percorrer a rede desde o vértice  $i$  e atingir o vértice  $j$ . Um caminho  $P(i/j)$  pode ser representado também pelo conjunto de vértices que o compõe, ou seja,  $P(i/j) = \{i, k, \dots, q, j\}$ .

A cada caminho  $P$ , parcial ou não, da rede pode-se associar um valor de tempo representando a duração do caminho  $P$ , denotada por  $T(P)$  e determinada pela soma das durações das atividades que compõem o caminho  $P$ , ou seja:  $T(P) = \sum_{(i,j) \in P} t(i, j)$ .

Um caminho crítico em um projeto é aquele de maior duração. A duração do projeto será determinada, portanto, pela duração de um caminho crítico. Um projeto, no entanto, pode ter mais de um caminho crítico.

A folga de um caminho  $P$ , denotada por  $F(P)$ , é a diferença entre a duração do projeto e a duração do caminho  $P$ .

Seja  $P(s/j) = \{P^1(s/j), P^2(s/j), \dots, P^p(s/j)\}$  o conjunto de caminhos parciais da rede com início no vértice  $s$  e término no vértice  $j$ . O caminho  $P^k(s/j) \in P(s/j)$  possui ordem  $k < p$  entre seus pares se  $\Pr[P^k(s/j) > P^{k+r}(s/j)] > \Pr[P^{k+r}(s/j) > P^k(s/j)]$ , para todo  $r = 1, 2, \dots, p - k$  e  $\Pr[\alpha]$  representa a probabilidade do argumento  $\alpha$ . Evidentemente, a definição aplica-se também a caminhos completos, isto é, para  $j = t$ .

### 3 O algoritmo de Dodin

O algoritmo de Dodin (1984) determina os caminhos k-críticos que dominam em probabilidade os demais caminhos da rede.

Considere-se a rede do projeto na forma canônica. Ao explorar um determinado vértice  $j$  da rede, o algoritmo identifica os caminhos parciais  $P(s/j)$  olhando apenas para os caminhos parciais dominantes  $P(s/v_i)$  determinados na iteração anterior, em que  $v_i$  são os vértices antecessores imediatos a  $j$ . Mantêm-se na lista de caminhos sempre aqueles  $k$  caminhos parciais  $P^i(s/v_i)$ ,  $j = 1, 2, \dots, k$  que dominam em probabilidade os demais caminhos que chegam em  $v_i$ . Percorre-se, assim, a rede a partir do vértice  $s$  e explora-se cada um dos nós da rede, até alcançar o vértice  $t$ .

Para compreender como o algoritmo de Dodin funciona, considere-se a rede da Figura 1 (que já se encontra na forma canônica), para a qual se desejam determinar os dois caminhos que dominam em probabilidade os demais. Desconsiderando-se os valores associados aos arcos da rede, inicie-se a aplicação do algoritmo pelo vértice  $s = 1$ . Considerando-se  $P^1(1/1)$  o caminho que incide sobre o nó 1, sobre o nó 2 incide apenas o caminho  $\{1, 2\} = [P^1(1/1) + (1, 2)]$ . Assim,  $P^1(1/2) = \{1, 2\}$ . Passa-se para o nó 3, sobre o qual incidem os caminhos  $\{1, 2, 3\} = [P^1(1/2) + (1, 3)]$  e  $\{1, 3\} = [P^1(1/1) + (1, 3)]$ . Supondo-se que o primeiro domine em probabilidade o segundo, denominamos  $P^1(1/3) = \{1, 2, 3\}$  e  $P^2(1/3) = \{1, 3\}$ . Passando para o vértice 4, verifica-se que os nós 1, 2 e 3 são antecessores imediatos a ele e, para determinar o caminho  $P^1(1/4)$ , confrontam-se os caminhos dominantes que chegam por esses nós, ou seja:  $\{1, 4\} = [P^1(1/1) + (1, 4)]$ ,  $[P^1(1/2) + (2, 4)]$  e  $[P^1(1/3) + (3, 4)]$ . Supondo-se que o dominante seja este último, tem-se  $P^1(1/4) = \{1, 2, 3, 4\}$ . O caminho  $P^2(1/4)$  é escolhido dentre os que restaram, isto é, caminhos  $[P^1(1/1) + (1, 4)]$ ,  $[P^1(1/2) + (2, 4)]$  e  $[P^2(1/3) + (3, 4)]$  (observe-se que  $P^1(1/3)$  já foi escolhido, então  $P^2(1/3)$  passa a ser o caminho dominante a partir do vértice 3). Suponha-se que  $P^2(1/4) = \{1, 2, 4\} = [P^2(1/2) + (2, 4)]$ . Finalmente, chega-se ao nó 5 que

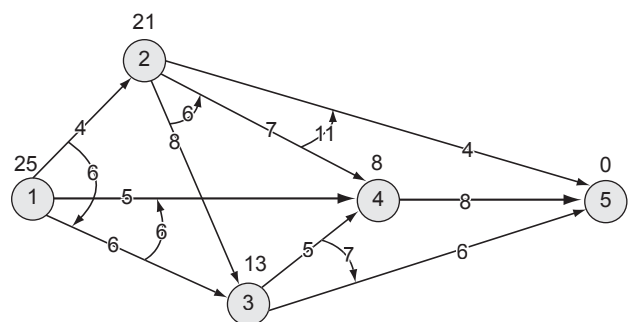


Figura 1. Rede do projeto para exemplificação dos algoritmos.

possui como antecessores imediatos os vértices 2, 3 e 4. Para se escolher  $P^1(1/5)$ , consideram-se apenas os caminhos dominantes que chegam por esses vértices, ou seja,  $[P^1(1/2) + (2, 5)]$ ,  $[P^1(1/3) + (3, 5)]$  e  $[P^1(1/4) + (4, 5)]$ . Suponha-se que  $P^1(1/5) = [P^1(1/4) + (4, 5)]$ . Em seguida, escolhe-se  $P^2(1/5)$  dentre  $[P^1(1/2) + (2, 5)]$ ,  $[P^1(1/3) + (3, 5)]$  e  $[P^2(1/4) + (4, 5)]$ .

O algoritmo de Dodin (1984) possui as seguintes características:

- a) a duração das atividades do projeto deve possuir distribuição de probabilidade discreta (é apresentada uma metodologia para discretizar a distribuição de probabilidade quando esta for contínua);
- b) identifica os  $k$  caminhos do projeto que dominam em probabilidade os demais caminhos;
- c) o procedimento apresentado possui complexidade computacional de ordem  $O(Ckn^2)$ , sendo  $C$  de ordem  $O([R]^2)$ , em que  $R$  é o número máximo de valores discretos aleatórios que a duração das atividades do projeto pode assumir,  $k$  é a quantidade de caminhos dominantes que se deseja,  $n$  é o número de nós da rede; e
- d) o procedimento apresentado nem sempre identifica corretamente os caminhos mais longos.

O Quadro 1 reproduz os resultados dos ensaios realizados por Dodin para a busca dos três caminhos dominantes sobre cada uma das redes ensaiadas. As redes estão identificadas pela dimensão do par de valores  $(V, A)$ , número de vértices e quantidade de arcos, respectivamente. Nos ensaios, foram identificados os verdadeiros três caminhos mais longos por enumeração completa, simulando a duração do projeto. Quando houve empate entre os caminhos identificados pelo seu algoritmo e pelo processo de simulação, foi lançado o valor 1 nas colunas relativas aos valores de  $k$ . Caso contrário, lançou-se o valor zero. Observa-se uma quantidade expressiva de redes, principalmente para valores maiores de  $k$ , nas quais a identificação não foi correta.

**Quadro 1.** Resultados dos ensaios apresentados por Dodin (1984).

Redes		Caminhos			Redes		Caminhos		
V	A	k = 1	k = 2	k = 3	V	A	k = 1	k = 2	k = 3
10	15	1	1	1	20	60	1	1	1
10	20	1	1	1	20	70	1	0	1
10	30	1	0	0	20	80	1	1	0
10	40	1	1	0	30	45	1	1	1
20	30	1	1	1	30	60	1	0	1
20	40	1	1	1	30	75	0	1	0
20	50	1	0	0	30	90	1	1	0

## 4 O algoritmo de Yen, Du e Ghanta

Yen et al. (1989) sugerem diversos algoritmos para identificar os  $k$  caminhos mais longos do projeto. O primeiro deles (denominado Algoritmo 1) é apresentado a seguir. Tanto a nomenclatura quanto a linguagem foram adaptadas para unificá-las com aquelas utilizadas neste artigo.

**Passo 1:** (Redução da rede original à forma canônica). Transforme a rede original do projeto na sua forma canônica, incluindo os vértices  $s = 1$  e  $t = n$  e os respectivos arcos fictícios.

**Passo 2:** (Rotulação dos vértices). Faça  $\bar{L}(n) = 0$  e determine, para cada um dos nós  $i$ ,  $i = n - 1, n - 2, \dots, 1$ ,  $\bar{L}(i) = \max_{v_i} (\bar{L}(v_i) + t(i, v_i))$ , em que  $v_i$  são todos os vértices sucessores imediatos de  $i$ .

**Passo 3:** (Criação dos  $k$  caminhos parciais). Crie  $k$  caminhos parciais  $P^i(s/v_i)$ ,  $i = 1, 2, \dots, k$ , em que  $s$  é o vértice inicial da rede e  $v_i$ ,  $i = 1, 2, \dots, k$  são os  $k$  nós sucessores imediatos de  $s$  que possuem os maiores valores de  $\bar{L}(v_i)$ . Se o número  $r_s$  de nós sucessores imediatos de  $s$  for menor que  $k$ , mantenha os caminhos parciais  $P^i(s/v_i)$ ,  $i = r_s + 1, \dots, k$  vazios.

**Passo 4:** (Rotulação dos  $k$  caminhos parciais). Determine para cada caminho parcial  $P^i(s/v_i)$ ,  $i = 1, 2, \dots, \min(k; r_s)$  o valor de  $T[P^i(s/v_i)] = [t(s, v_i) + \bar{L}(v_i)]$  e disponha os caminhos parciais  $P^i(s/v_i)$  em ordem não crescente dos valores de  $T[P^i(s/v_i)]$ . Para os caminhos vazios faça  $T[P^i(s/v_i)] = 0$ . Assim,  $P^1(s/v_i)$  será o caminho parcial com origem no vértice  $s$  e término no vértice  $v_i$  sucessor imediato de  $s$  que possui o maior valor de  $T[P^i(s/v_i)]$ , para  $i = 1, 2, \dots, \min(k; r_s)$ .

**Passo 5:** (Ramificação de um caminho parcial). Faça  $i = 1$ . Seja  $q$  o último vértice incluído no caminho  $P^i(s/q)$ . Se  $q = t$ , siga para o passo 9. Caso contrário, ramifique o caminho  $P^i(s/q)$  em  $n$  novos caminhos,  $P_r^i(s/u_r)$ , em que  $n$  é o número de nós  $u_r$ ,  $r = 1, 2, \dots, n$ , sucessores imediatos de  $q$ . Calcule  $T[P_r^i(s/u_r)] = [ \sum_{(i,j) \in P^i(s,u_r)} t(i,j) + \bar{L}(u_r) ]$ ,  $r = 1, 2, \dots, n$  e disponha os  $n$  novos caminhos  $P_r^i(s/u_r)$  em ordem não crescente dos valores de  $T[P_r^i(s/u_r)]$ .

**Passo 6:** (Extensão de um caminho parcial). Estenda o caminho parcial  $P^i(s/q)$  incluindo o nó  $u_r$ ,  $r = 1, 2, \dots, n$  adjacente a  $q$  com o maior valor de  $T[P_r^i(s/u_r)]$ . Esse caminho será  $P_1^i(s/u_1)$ .

**Passo 7:** (Substituição de caminhos parciais). Se  $T[P_2^i(s/u_0)] > T[P^i(s/v_i)]$  para algum caminho  $P^i(s/v_i)$ ,  $j = i + 1, \dots, k$ , parcial ou não, substitua esse caminho parcial por  $P_2^i(s/u_2)$ . Proceda de forma análoga para os demais caminhos parciais  $P_r^i(s/u_r)$ ,  $r = 2, \dots, n$ , ou seja, enquanto existirem caminhos parciais  $P^i(s/v_i)$ ,  $j = i + 1, \dots, k$ , com  $T[P^i(s/v_i)] < T[P_r^i(s/u_r)]$ , para algum  $r = 2, \dots, n$ , substitua  $P^i(s/v_i)$  por  $P_r^i(s/u_r)$ .

**Passo 8:** (Obtenção do  $i$ -ésimo caminho mais longo). Repita os passos de 5 a 7 até que  $P^i(s/q)$  se torne um caminho completo, que ocorrerá quando o vértice  $q$  for antecessor imediato do vértice  $t$ . Nesse momento,  $P_1^i(s/u_1) = P_1^i(s/t) = P^i$  e  $T(P^i) = T[P_1^i(s/t)]$ .

**Passo 9:** (Construção dos demais caminhos mais longos). Se  $i = k$ , pare. Caso contrário, faça  $i = i + 1$  e retorne ao passo 5.

Aplicando-se o procedimento à rede da Figura 1, verifica-se que ela já se encontra na forma canônica, com  $s = 1$  e  $t = 5$ . Adotando-se o procedimento PERT determina-se  $L(5) = E(5) = 25$ ,  $L(4) = 17$ ,  $L(3) = 12$ ,  $L(2) = 4$  e  $L(1) = 0$ . Assim, os valores de  $\bar{L}(i)$  serão iguais a 25, 21, 13, 8 e 0 para os nós  $i = 1, 2, 3, 4$  e 5, respectivamente. Supondo-se que se queira determinar os 4 caminhos mais longos do projeto ( $k = 4$ ), pelo passo 4 do algoritmo, teremos  $P^1(1/2) = \{1, 2\}$ ,  $P^2(1/3) = \{1, 3\}$ ,  $P^3(1/4) = \{1, 4\}$ ,  $P^4 = \{\emptyset\}$ , com  $T[P^i(s/v_i)]$  iguais a 25, 19, 13 e 0, para  $i = 1, 2, 3, 4$ , respectivamente. Pelo passo 5, o caminho parcial  $P^1(1/2)$  é ramificado em três novos caminhos (existem 3 nós adjacentes ao nó  $q = 2$ ,  $u_1 = 3$ ,  $u_2 = 4$  e  $u_3 = 5$ ):  $P_1^1(1/3) = \{1, 2, 3\}$ ,  $P_2^1(1/4) = \{1, 2, 4\}$  e  $P_3^1(1/5) = \{1, 2, 5\}$ , com os valores de  $T[P_r^1(s/u_r)]$ , respectivamente iguais a 25, 19 e 8, listados já em ordem não crescente dos valores de  $T[P_r^1(s/u_r)]$ . Pelo passo 6, o caminho  $P^1$  é estendido pela inclusão do vértice 3, passando a  $P^1(s/3) = \{1, 2, 3\}$ . Aplicando-se o passo 7, verifica-se que  $T[P_2^1(1/4)] = 19$  é maior que  $T[P^3(1/4)] = 13$ . Assim, o caminho original  $P^3(1/4)$  é substituído por  $P_2^1(1/4)$ ; ou seja,  $P^3(1/4) = \{1, 2, 4\}$ . Da mesma forma,  $T[P_3^1(1/5)] = 8$  é maior que  $T[P^4] = 0$ . Assim,  $P^4$  é substituído por  $P_3^1(1/5)$ , ou seja,  $P^4(1/5) = \{1, 2, 5\}$ . O caminho  $P^2(1/3)$  permanece o mesmo.

Reaplicando-se os passos de 5 a 7, o caminho parcial  $P^1(1/3)$  é ramificado em dois novos caminhos:  $P_1^1(1/4) = \{1, 2, 3, 4\}$  e  $P_2^1(1/5) = \{1, 2, 3, 5\}$ , com os valores de  $T[P_r^1(1/u_r)]$ , respectivamente iguais a 25 e 18, já ordenados, portanto. Assim, o caminho  $P^1(1/3)$  é estendido pela inclusão do vértice 4, passando a  $P^1(1/4) = \{1, 2, 3, 4\}$ . Como  $T[P_2^1(1/5)] = 18$  é maior que  $T[P^4(1/5)] = 8$ , o atual caminho  $P^4(1/5)$  é substituído por  $P_2^1(1/5)$ , ou seja,  $P^4(1/5) = \{1, 2, 3, 5\}$ . Reaplicando-se os passos de 5 a 7 novamente, o caminho parcial  $P^1(1/4) = \{1, 2, 3, 4\}$  é estendido pela inclusão do vértice  $t = 5$  (único adjacente ao último nó incluído em  $P^1(s/q)$ ,  $q = 4$ ), tornando-se o primeiro caminho mais longo do projeto, ou seja,  $P^1 = \{1, 2, 3, 4, 5\}$ , com duração igual a  $T(P^1) = 25$ . Os demais caminhos são  $P^2(1/3) = \{1, 3\}$ ,  $P^3(1/4) = \{1, 2, 4\}$  e  $P^4(1/5) = \{1, 2, 3, 5\}$ , com valores de  $T[P^i(s/u_i)]$  iguais a 19, 19 e 18, respectivamente.

O segundo caminho mais longo do projeto é construído a partir de  $P^2(1/3) = \{1, 3\}$ . Os nós adjacentes ao vértice  $q = 3$  são  $u_1 = 4$  e  $u_2 = 5$ . Assim, o caminho  $P^2(1/3)$  é ramificado em  $P_1^2(1/4) = \{1, 3, 4\}$  e em  $P_2^2(1/5) = \{1, 3, 5\}$ . Com isso, pela absorção do vértice 4 (maior valor de

$T[P_1^2(1/u_r)]$ ), tem-se  $P^2(1/4) = \{1, 3, 4\}$ , com  $T[P^2(1/4)] = 19$ . O caminho  $P_2^2(1/5)$ , com  $T[P_2^2(1/5)] = 12$ , não substitui nenhum outro caminho. Na próxima ramificação de  $P^2(1/4)$ , ele absorve o nó 5 (único adjacente ao vértice  $q = 4$ ) e se transforma no segundo caminho mais longo do projeto,  $P^2 = \{1, 3, 4, 5\}$ , com  $T(P^2) = 19$ . O terceiro caminho mais longo,  $P^3 = \{1, 2, 4, 5\}$ , com duração igual a 19, é obtido pela absorção do vértice 5, a partir da expansão do caminho parcial  $P^3(1/4) = \{1, 2, 4\}$ . Explorando-se o último caminho em construção, tem-se  $P^4(1/5) = \{1, 2, 3, 5\}$  e como  $5 = t$ , vai-se ao passo 9 e o processo se encerra, obtendo  $P^4 = \{1, 2, 3, 5\}$  com duração igual a 18.

Para melhorar a eficiência desse algoritmo, é proposto um procedimento, reproduzido a seguir, cujo objetivo é reduzir o número de ramificações que são feitas conforme o Passo 5 do Algoritmo 1, originando o Algoritmo 1 Melhorado.

Seja  $q$  o último vértice incluído no caminho em expansão  $P^i(s/q) = (s, v_1, v_2, \dots, q)$  e considere a lista com os vértices  $u_r$  sucessores imediatos de  $q$  ordenados segundo os valores não crescentes de  $T[P_r^i(s/u_r)]$ .

**Passo 1:** Crie os caminhos  $P_1^i(s/u_1)$  e  $P_2^i(s/u_2)$  incluindo em  $P^i(s/q)$  o primeiro e o segundo vértices da lista ordenada de vértices sucessores imediatos de  $q$ , respectivamente.

**Passo 2:** Faça  $r = 2$ . Se existir algum caminho  $P^i(s/v_j)$ ,  $j = i + 1, \dots, k$ , parcial ou não, tal que  $T[P_r^i(s/u_r)] > T[P^i(s/v_j)]$ , substitua esse caminho por  $P_r^i(s/u_r)$  e crie o caminho  $P_{r+1}^i(s/u_{r+1})$ . Caso contrário, pare.

**Passo 3:** Faça  $r = r + 1$  e retorne ao passo 2.

O Algoritmo 1 Melhorado mostrou-se progressivamente mais eficiente do que o algoritmo original, à medida que o número  $k$  de caminhos cresce. Para  $k = 100$ , o tempo de execução do algoritmo foi reduzido em cerca de 60%, conforme mostram os experimentos desenvolvidos por Yen et al. (1989).

São fornecidos ainda alguns outros algoritmos, cuja eficiência, contudo, não se mostrou significativamente superior a desse último, conforme ensaios realizados.

## 5 O algoritmo de Ju e Saleh

Em Ju e Saleh (1991), pode-se encontrar o algoritmo mais eficiente que a literatura fornece para determinar os caminhos  $k$ -críticos em redes direcionadas e não cíclicas. Observou-se, contudo, que os algoritmos apresentados por Yen et al. (1989) mostram-se pouco eficientes para valores grandes de  $k$ . Isto ocorre devido às ramificações que devem ser feitas após a extensão de um caminho parcial pela inclusão de um novo vértice. Para evitar esse inconveniente, definiu-se, para cada par de arcos adjacentes  $(i, j)$  e  $(i, k)$  da rede  $G(V, A)$ , um tipo de folga, a qual foi chamada de folga relativa do arco  $(i, j)$  em

relação ao arco  $(i, k)$ , denominada  $fr[(i,j)/(i,k)]$ , dada pela diferença  $f(i, k) - f(i, j)$ , desde que  $f(i, k) > f(i, j)$ . Obteve-se, inicialmente, o primeiro caminho mais longo, facilmente identificável, e ramificou-se para fora desse caminho, na sua aresta de menor folga relativa, para se obter o segundo caminho mais longo da rede. Repetiu-se o processo sobre a aresta de segunda menor folga relativa e assim sucessivamente, até que todas as folgas relativas das arestas do caminho mais longo tivessem sido utilizadas. A partir daí, passou-se a ramificar o segundo caminho mais longo utilizando-se suas folgas relativas, e assim por diante, até que fossem identificados os  $k$  caminhos mais longos desejados.

Os passos seguintes formalizam a implementação do algoritmo de Ju e Saleh (1991). Tanto a nomenclatura quanto a linguagem foram adaptadas para unificá-las com aquelas utilizadas neste artigo.

**Passo 1:** (Redução da rede original à forma canônica). Transforme a rede original do projeto na sua forma canônica, incluindo os vértices  $s = 1$  e  $t = n$  e os respectivos arcos fictícios.

**Passo 2:** (Rotulação dos vértices). Faça  $\bar{L}(n) = 0$  e determine, para cada um dos nós  $i$ ,  $i = n - 1, n - 2, \dots, 1$ ,  $\bar{L}(i) = \max_{v_i} (\bar{L}(v_i) + t(i, v_i))$ , em que  $v_i$  são todos os vértices sucessores imediatos de  $i$ .

**Passo 3:** (Identificação do caminho mais longo da rede). Dentre os vértices  $v_i$ ,  $i = 1, 2, \dots, r$ , sucessores imediatos de  $s$ , inclua aquele que possui o maior valor de  $c(v_i) = [t(s, v_i) + \bar{L}(v_i)]$ . Seja  $u$  esse vértice. Faça  $s = u$  e repita o procedimento até incluir o vértice  $t$ . Seja  $P^1$  o caminho determinado.

**Passo 4:** (Determinação das folgas relativas). Para cada vértice  $u$  da rede, rotule seus sucessores imediatos  $v_i$ ,  $i = 1, 2, \dots, r$ , com os valores de  $c(v_i) = t(u, v_i) + \bar{L}(v_i)$  e ordene os vértices  $v_i$  em ordem não crescente de valores de  $c(v_i)$ , originando, para cada vértice  $u$  da rede, a lista ordenada de vértices sucessores  $v_1, v_2, \dots, v_r$ . Determine a folga relativa do arco  $(u, v_i)$  em relação ao arco  $(u, v_{i+1})$ ,  $f_r[(u, v_i)/(u, v_{i+1})]$ , pela expressão  $f_r[(u, v_i)/(u, v_{i+1})] = c(v_i) - c(v_{i+1})$ ,  $i = 1, 2, \dots, r - 1$ , para todo vértice  $u$  da rede. O arco  $(u, v_{i+1})$  será chamado de arco adjacente imediato ao arco  $(u, v_i)$ ,  $i = 1, 2, \dots, r - 1$ . Assim, a folga de um arco será definida apenas em relação ao seu arco adjacente imediato.

**Passo 5:** (Ramificações de  $P^i$ ). Considerando  $P^i$  o último caminho identificado, para cada arco  $(u, v)$  do caminho  $P^i$ , liste seu arco adjacente imediato  $(u, v_i)$ , se existir, e ordene os arcos listados em ordem não decrescente dos valores da folga relativa, criando a lista,  $L(\text{arcos}(P^i))$ , de arcos  $(u, v_i)$ ,  $i = 1, 2, \dots, p$ , em que  $p$  é o número de arcos do caminho  $P^i$  que possui arcos adjacentes imediatos. Para que se evite enumerar um mesmo caminho duas vezes, não inclua na lista  $L(\text{arcos}(P^i))$  os arcos  $(i, j)$  já utilizados na formação de um caminho anterior  $P^j$ ,  $j < i$ . Denote por  $T[\text{Prox}(P^i)]$  a duração do próximo caminho mais longo que se ramificará de  $P^i$ , a qual será calculada por  $T[\text{Prox}(P^i)] = T(P^i) - f_r(u, v_i)_1$ , em que  $(u, v_i)_1$  será o primeiro arco da lista  $L(\text{arcos}(P^i))$ .

**Passo 6:** (Identificação do próximo caminho mais longo). Dentre todos os caminhos  $P^i$  já identificados,  $i = 1, 2, \dots, n < k$ , considere  $P^*$  aquele que possui o maior valor de  $T[\text{Prox}(P^i)]$ . Construa o próximo caminho mais longo da rede,  $P^{n+1}$ , da seguinte maneira: siga ao longo do caminho  $P^*$ , ramifique para fora desse caminho no primeiro arco da lista  $L(\text{arcos}(P^*))$  e complete o caminho  $P^{n+1}$  utilizando o procedimento contido no passo 3. Remova o primeiro arco da lista  $L(\text{arcos}(P^*))$  e atualize o valor de  $T[\text{Prox}(P^*)]$ , se  $L(\text{arcos}(P^*)) = \emptyset$ , faça  $T[\text{Prox}(P^*)] = 0$ .

**Passo 7:** (Identificação dos demais caminhos mais longos). Repita os passos 5 e 6 até que os  $k$  caminhos desejados sejam listados.

Passemos a considerar a aplicação do algoritmo de Ju e Saleh (1991) à rede da Figura 1 para obtenção dos cinco caminhos mais longos, sendo que a rede já se encontra na forma canônica, com  $s = 1$  e  $t = 5$ . Os valores de  $\bar{L}(i)$  são iguais a 25, 21, 13, 8 e 0, para os nós  $i = 1, 2, 3, 4$  e 5, respectivamente. O caminho mais longo da rede é  $P^1 = \{1, 2, 3, 4, 5\}$ . A rotulação dos vértices sucessores de  $u$  fornece: para  $u = 1$ ,  $c(2) = 25$ ,  $c(3) = 19$  e  $c(4) = 13$ ; para  $u = 2$ ,  $c(3) = 21$ ,  $c(4) = 15$  e  $c(5) = 4$ ; para  $u = 3$ ,  $c(4) = 13$  e  $c(5) = 6$ ; e, para  $u = 4$ ,  $c(5) = 8$ . Como todas as listas já estão ordenadas, pode-se calcular as folgas relativas em cada vértice (Figura 1): para  $u = 1$ ,  $f_r[(1, 2)/(1, 3)] = 25 - 19 = 6$  e  $f_r[(1, 3)/(1, 4)] = 19 - 13 = 6$ ; para  $u = 2$ ,  $f_r[(2, 3)/(3, 4)] = 21 - 15 = 6$  e  $f_r[(2, 4)/(2, 5)] = 15 - 4 = 11$ ; e, para  $u = 3$ ,  $f_r[(3, 4)/(3, 5)] = 13 - 6 = 7$ . As explicações a seguir podem ser acompanhadas observando-se a Tabela 1.

**Tabela 1.** Evolução do algoritmo de Ju e Saleh (1991).

Caminho ( $P^i$ )	Arco ramificador	Caminho (duração)	$T[\text{Prox}(P^i)]$	$L(\text{arcos}(P^i))/f_r$
$P_1$		{1, 2, 3, 4, 5} (25)	19, 19, 18, 0	(1,3)/6; (2, 4)/6; (3, 5)/7
$P_2$	(1, 3) $\in P_1$	{1, 3, 4, 5} (19)	13, 12	(1, 4)/6; (3, 5)/7
$P_3$	(2, 4) $\in P_1$	{1, 2, 4, 5} (19)	8	(2, 5)/11
$P_4$	(3, 5) $\in P_1$	{1, 2, 3, 5} (18)	7	(2, 5)/11
$P_5$	(1, 4) $\in P_2$	{1, 4, 5} (13)		

O último caminho identificado é o caminho mais longo do projeto,  $P^1 = \{1, 2, 3, 4, 5\}$ , com duração  $T(P^1) = 25$ . Os arcos  $(u, v)$  de  $P^1$  são  $(1, 2)$ ,  $(2, 3)$ ,  $(3, 4)$  e  $(4, 5)$ , cujos arcos adjacentes imediatos são, respectivamente,  $(1, 3)$ ,  $(2, 4)$  e  $(3, 5)$ , com folgas relativas respectivamente iguais a 6, 6, 7. Assim,  $L(\text{arcos}(P^1)) = \{(1, 3), (2, 4) \text{ e } (3, 5)\}$ . O próximo caminho mais longo  $P^2$ , ramificará de  $P^* = P^1$  no arco  $(1, 3)$  e, aplicando-se o procedimento do Passo 3 a partir do vértice 3, são incluídos os vértices 4 e 5, originando-se  $P^2 = \{1, 3, 4, 5\}$ , com duração  $T(P^2) = 19$ . Excluindo-se o arco  $(1, 3)$  da lista  $L(\text{arcos}(P^1))$ , tem-se  $T[\text{Prox}(P^1)] = 25 - 6 = 19$ . Por sua vez,  $L(\text{arcos}(P^2)) = \{(1, 4), (3, 5)\}$ , com folgas relativas respectivamente iguais a 6 e 7. Portanto,  $T[\text{Prox}(P^2)] = 19 - 6 = 13$  e  $P^* = P^1$ . Assim, o caminho  $P^3$  ramificará também de  $P^1$  no arco  $(2, 4)$ , originando o caminho  $P^3 = \{1, 2, 4, 5\}$ , com  $T(P^3) = 19$ . Ramificando-se esse caminho, obtém-se  $L(\text{arcos}(P^3)) = \{(2, 5)\}$  e  $T[\text{Prox}(P^3)] = 19 - 11 = 8$ . Observe-se que o arco  $(1,3)$  já foi utilizado para gerar o caminho  $P^2$  e, portanto, não integra a lista  $L(\text{arcos}(P^3))$ . Removendo-se o arco  $(2, 4)$  de  $L(\text{arcos}(P^1))$ , obtém-se  $T[\text{Prox}(P^1)] = 25 - 7 = 18$ , que é ainda o maior dos valores de  $T[\text{Prox}(P^i)]$ ,  $i = 1, 2$  e  $3$ . Assim, o caminho  $P^4$  ramificará de  $P^* = P^1$  no arco  $(3, 5)$ , originando o caminho  $P^4 = \{1, 2, 3, 5\}$ , com  $T(P^4) = 18$ . Ramificando-se esse caminho, obtém-se  $L(\text{arcos}(P^4)) = \{(2, 5)\}$  e  $T[\text{Prox}(P^4)] = 18 - 11 = 7$ . Removendo-se o arco  $(3, 5)$  de  $L(\text{arcos}(P^1))$ , este conjunto torna-se vazio e, portanto, faz-se  $T[\text{Prox}(P^1)] = 0$ . Assim, o próximo e último caminho desejado,  $P^5$ , ramificará de  $P^2$  (é o que possui o maior valor de  $T[\text{Prox}(P^i)]$ ,  $i = 1, 2, 3$  e  $4$ ) por meio do arco  $(1, 4)$ , originando  $P^5 = \{1, 4, 5\}$  com duração  $T(P^5) = 13$ .

Foram realizados testes computacionais para demonstrar a melhor eficiência do algoritmo proposto, comparativamente àqueles apresentados por Yen et al. (1989), obtendo-se redução no tempo de processamento computacional da ordem de 50%.

## 6 O algoritmo da Folga Mínima

A principal contribuição deste artigo é o desenvolvimento do algoritmo da folga mínima para determinar os caminhos k-críticos numa rede PERT, mais simples do que aqueles anteriormente apresentados. Baseia-se na percepção de que a folga  $f(i, j)$  da atividade  $(i, j)$  do projeto é dada pela diferença entre a duração do projeto,  $T$ , e a duração do caminho mais longo que passa por essa atividade. Assim, se existir uma atividade  $(i, j)$  no projeto com folga  $f(i, j)$ , então o caminho mais longo que passa por essa atividade terá duração dada por  $T - f(i, j)$ . As proposições 1 e 2, apresentadas a seguir, constituem a base teórica para a validação do algoritmo da folga mínima.

**Proposição 1:** considerando-se a rede do projeto na forma canônica, sendo  $s$  e  $t$  o primeiro e o último vértices, respectivamente, e  $T = E(t)$  a duração do projeto, o caminho mais longo que passa pela atividade  $(p, q)$  possui duração igual a  $T - f(p, q)$ , em que  $f(p, q)$  é a folga da atividade  $(p, q)$ .

**Demonstração:** a data mais cedo do vértice  $p$ ,  $E(p)$ , é igual à duração do caminho mais longo entre o vértice  $s$  e o vértice  $p$ . Por outro lado,  $[T - L(p)]$  é igual à duração do caminho mais longo entre os vértices  $p$  e  $t$ . Considerando-se  $P^k$  o caminho mais longo que passa pela atividade  $(p, q)$  e  $t(p, q)$  e  $f(p, q)$  a duração e a folga de  $(p, q)$ , respectivamente, então, a duração de  $P^k$ ,  $T(P^k)$ , é dada por  $T(P^k) = E(p) + t(p, q) + [T - L(p)]$ ; como  $f(p, q) = L(q) - E(p) - t(p, q)$ , então  $T(P^k) = T - f(p, q)$ .

Para compreender o funcionamento do algoritmo da folga mínima, suponha-se que se deseje determinar o segundo caminho mais longo em um dado projeto. Inicialmente, localiza-se uma atividade com a segunda menor folga do projeto, que servirá de semente para a construção do caminho desejado. Considerando-se  $(i, j)$  essa atividade, a partir dela, identifica-se, dentre todas as atividades sucessoras imediatas a  $(i, j)$ , aquela que possui a menor folga. Considerando-se  $(j, k)$  essa atividade, busca-se então a atividade sucessora imediata de  $(j, k)$  com a menor folga e assim, sucessivamente, por meio de um processo *forward*, até que a última atividade identificada conduza ao vértice  $t$ , construindo o caminho parcial  $P(j/t)$ . De forma idêntica, por um processo *backward*, a partir do nó  $i$  da atividade semente  $(i, j)$ , constrói-se o caminho parcial  $P(s/i)$ . O segundo caminho mais longo do projeto será então formado pela união da atividade  $(i, j)$  com os dois caminhos parciais  $P(s/i)$  e  $P(j/t)$ .

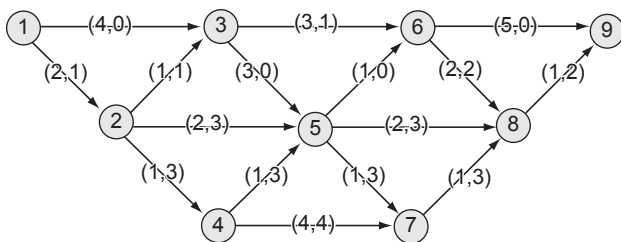
**Proposição 2:** Os processos *forward* e *backward* a partir da atividade semente  $(p, q)$ , percorrendo as atividades de menores folgas da rede, levam à identificação do caminho mais longo do projeto passando por  $(p, q)$ .

**Demonstração:** Considerando-se: a rede do projeto na forma canônica, em que  $s$  e  $t$  são o primeiro e o último vértices, respectivamente;  $(p, q)$  uma atividade semente para construção do caminho mais longo,  $P$ , que passa por  $(p, q)$ ;  $u$  e  $v$  dois nós sucessores imediatos de  $q$ , com  $f(q, u) < f(q, v)$ ;  $i$  e  $j$ , dois nós antecessores imediatos de  $p$ , com  $f(i, p) < f(j, p)$ ; e  $T$ , a duração do projeto,  $T = E(t)$ , o parâmetro  $\bar{L}(i) = T - L(i)$  de um vértice  $i$  qualquer da rede é igual à duração do maior caminho entre os vértices  $i$  e  $t$ . Então, uma nova atividade  $(q, u)$  sucessora imediata de  $(p, q)$ , a ser incorporada pelo processo *forward* ao caminho  $P$  em construção, deve ser tal que  $\bar{L}(u) + t(q, u) = \bar{L}(q)$ . Se  $f(q, u) < f(q, v)$  então  $T - f(q, u) > T - f(q, v)$ , uma vez que  $T > f(i, j)$  para qualquer atividade  $(i, j)$  do projeto. Como  $f(i, j) = L(j) - E(i) - t(i, j)$ , para qualquer  $(i, j)$ ,  $T - [L(u) - E(q) - t(q, u)] > T - [L(v) - E(q) - t(q, v)]$ , o que fornece  $\bar{L}(u) + t(q, u) > \bar{L}(v) + t(q, v)$ , indi-

quando que  $L(q)$  é determinado a partir do vértice  $u$ , ou seja  $L(q) = L(u) - t(q, u)$ . Logo,  $L(u) = L(q) + t(q, u)$  e  $T - L(u) = T - [L(q) + t(q, u)]$ , o que fornece  $\bar{L}(u) = T - L(q) - t(q, u)$  ou  $\bar{L}(u) + t(q, u) = \bar{L}(q)$ . Considerando-se agora o processo *backward*, uma nova atividade  $(i, p)$  antecessora imediata de  $(p, q)$ , a ser incorporada ao caminho  $P$  em construção, deve ser tal que  $E(i) + t(i, p) = E(p)$ , uma vez que  $E(p)$  é igual a duração do caminho mais longo entre os vértices  $s$  e  $i$ . Se  $f(i, p) < f(j, p)$ , então  $L(p) - E(i) - t(i, p) < L(p) - E(j) - t(j, p)$  e  $E(i) + t(i, p) > E(j) + t(j, p)$ . Assim,  $E(p)$  é determinado a partir do vértice  $i$ , ou seja,  $E(p) = E(i) + t(i, p)$ .

Para exemplificar, considere-se a rede da Figura 2, em que  $s = 1$  e  $t = 9$ , sobre a qual se deseja determinar os caminhos com duração igual ou maior a  $T^* = 11$ . Os dois valores associados a cada arco da rede representam a duração e a folga da atividade, respectivamente. O caminho mais longo,  $P^1$ , passa por uma atividade de folga zero. Escolhendo a atividade  $(i, j) = (1, 3)$  como semente, pelo processo *forward*, incluem-se as atividades  $(3, 5)$ ,  $(5, 6)$  e  $(6, 9)$ , uma após a outra, formando o caminho parcial  $P^1(j/t) = P^1(3/9)$  (são as atividades com menores folga no percurso entre os vértices  $j = 3$  e  $t = 9$ ). Como a atividade semente tem origem no vértice inicial da rede, não há necessidade de aplicar o processo *backward* e, o caminho mais longo  $P^1$  é o formado pela união de  $(1, 3)$  com  $P^1(j/t)$ , ou seja,  $P^1 = \{1, 3, 5, 6, 9\}$ , com duração igual a 13.

Para identificar o próximo caminho mais longo, busca-se uma atividade com a menor folga, desconsiderando-se as atividades pertencentes ao caminho  $P^1$ . Encontra-se, por exemplo, a atividade  $(2, 3)$ , com  $f(2, 3) = 1$ . Assim, por essa atividade passa-se um caminho com duração igual a 12. Pelo processo *forward*, incluem-se as atividades  $(3, 5)$ ,  $(5, 6)$  e  $(6, 9)$  e pelo processo *backward* inclui-se a atividade  $(1, 2)$ , obtendo-se o segundo caminho mais longo  $P^2 = \{1, 2, 3, 5, 6, 9\}$ , com duração igual a 12. Desconsiderando-se também as atividades desse caminho, verifica-se que a atividade de menor folga é  $(3, 6)$ , com  $f(3, 6) = 1$ . Utilizando-se essa mesma atividade como semente para o próximo caminho, obtém-se  $P^3 = \{1, 3, 6, 9\}$ , com duração também igual a 12. Como



**Figura 2.** Rede para desenvolvimento do Algoritmo da folga mínima.

existem as atividades  $(6, 8)$  e  $(8, 9)$  com folga igual a 2, existe pelo menos um caminho com duração igual a 11. Escolhendo-se  $(8, 9)$  como semente, encontra-se  $P^4 = \{1, 3, 5, 6, 8, 9\}$ . Desconsiderando-se todas as atividades presentes nos caminhos já identificados, a menor folga de atividade que aparece na rede é igual a 3. Assim, o próximo caminho mais longo terá duração igual a  $10 < T^*$ , e pode-se encerrar o processo de busca.

O principal propósito da busca de caminhos no gerenciamento de projetos é identificar os caminhos que possuem duração maior do que um dado valor  $T^*$ . O algoritmo da folga mínima é apresentado com esse propósito. Com pequenas modificações, pode também ser utilizado para identificar os  $k$  caminhos mais longos do projeto.

A seguir, formaliza-se o procedimento exemplificado acima, ao qual chamaremos de procedimento de busca de caminhos explícitos pela folga.

### 6.1 Procedimento de busca de caminhos explícitos pela folga

Para identificar os caminhos com duração igual ou superior a  $T^*$ , aplica-se o seguinte procedimento:

**Passo 1:** (Redução da rede original à forma canônica). Transforme a rede original do projeto na sua forma canônica, incluindo os vértices  $s = 1$  e  $t = n$  e os respectivos arcos fictícios.

**Passo 2:** (Determinação das folgas das atividades). Faça  $E(s) = 0$  e determine as datas mais cedo,  $E(i)$ , para os eventos  $i = 2, 3, \dots, n$ ; faça  $L(t) = E(t)$  e determine as datas mais tarde,  $L(i)$ , para os eventos  $i = t-1, t-2, \dots, 1$ ; e determine as folgas  $f(i, j)$  para cada atividade  $(i, j)$  do projeto pela expressão  $f(i, j) = L(j) - E(i) - t(i, j)$ , em que  $t(i, j)$  é a duração da atividade  $(i, j)$ . Crie a lista  $L[(i, j)]$  com as atividades do projeto listadas em ordem não crescente dos valores de suas folgas, incluindo apenas as atividades  $(i, j)$  tais que  $f(i, j) \leq [L(t) - T^*]$ .

**Passo 3:** (Identificação da atividade semente). Considere  $k$  o índice do caminho  $P^k$ . Faça  $k = 1$ . Identifique uma atividade com a menor folga na lista  $L[(i, j)]$ . Considere  $(p, q)$  essa atividade.

**Passo 4:** (Identificação da necessidade do processo *forward*). Se  $q = t$ , faça  $P^k(q/t) = \emptyset$  e siga para o Passo 7.

**Passo 5:** (Processo *forward* entre os vértices  $q$  e  $t$  - caminho parcial  $P^k(q/t)$ ). Faça  $p^k(q/t) = \emptyset$ . Dentre as atividades  $(q, v_i)$ ,  $i = 1, 2, \dots, m$ , sucessoras imediatas de  $(p, q)$ , inclua no caminho  $p^k(q/t)$  aquela que possui a menor folga. Considere  $(q, v)$  essa atividade. Faça  $(p, q) = (q, v)$  e repita o procedimento, posicionando no caminho  $P^k(q/t)$  cada nova atividade após a última incluída, até incluir a atividade  $(q, t)$ .

**Passo 6:** (Identificação da necessidade do processo *backward*). Se  $p = s$ , faça  $P^k(s/p) = \emptyset$  e siga para o Passo 8.



**Passo 7:** (Processo *backward* entre os vértices  $s$  e  $p$  – caminho parcial  $P^k(s/p)$ ). Faça  $P^k(s/p) = \emptyset$ . Dentre as atividades  $(r_i, p)$ ,  $i = 1, 2, \dots, n$ , antecessoras imediatas de  $(p, q)$ , inclua em  $P^k(s/p)$  aquela que possui a menor folga. Seja  $(r, p)$  essa atividade. Faça  $(p, q) = (r, p)$  e repita o procedimento, posicionando no caminho  $P^k(q/t)$  cada nova atividade antes da última incluída, até incluir a atividade  $(s, p)$ .

**Passo 8:** (Identificação do caminho  $P^k$ ). Identifique o  $k$ -ésimo caminho mais longo do projeto por  $P^k = P^k(s/p) + (p, q) + P^k(q/t)$ .

**Passo 9:** (Atualização da lista de atividades). Elimine da lista  $L[(i, j)]$  as atividades pertencentes a  $P^k$ , faça  $k = k + 1$  e retorne ao Passo 3.

Se, durante a aplicação do processo *forward* (*backward*) existir, a partir de um vértice  $q$  (ou  $p$ ), mais de uma atividade sucessora (ou antecessora) imediata com a mesma menor folga, deve-se completar o processo, retornar ao vértice  $(q, p)$  e repeti-lo, escolhendo desta vez uma outra atividade com a menor folga.

O procedimento proposto é mais simples do que aquele apresentado por Ju e Saleh (1991). Não há necessidade de determinar as folgas relativas das atividades (Passo 3 do algoritmo de Ju e Saleh), o que é feito por um processo enumerativo, pois exige que se faça uma ordenação de todos os vértices  $v_i$  sucessores imediatos de cada um dos vértices  $u$  da rede. Evita-se também a necessidade de identificar as ramificações cada vez que um caminho  $P^i$  é identificado (Passo 4 do algoritmo de Ju e Saleh), outro processo enumerativo.

O procedimento de busca de caminhos explícitos pela folga pode, no entanto, não revelar todos os caminhos desejados do projeto. Se um caminho  $P^k$  é totalmente recoberto por atividades de dois ou mais caminhos de maior duração que  $P^k$ , esse caminho não será revelado pelo procedimento proposto. É o que ocorre com o caminho  $\{1, 2, 3, 6, 9\}$  da rede da Figura 2, que possui duração igual a 11. Este caminho é recoberto pelos caminhos  $\{1, 2, 3, 5, 6, 9\}$  e  $\{1, 3, 6, 9\}$ , ambos com duração 12. Note-se, porém, que os caminhos não explícitos pela folga mínima são caminhos de menor duração do projeto e podem não estar entre aqueles que são necessários identificar, quando do gerenciamento do projeto. Para avaliar a incidência desses caminhos em redes PERT, foram implementados testes computacionais, conforme exposto na seção seguinte.

Para identificar os caminhos não explícitos pela folga mínima, é necessário executar um segundo procedimento, ao qual chamaremos de *procedimento de busca de caminhos implícitos*, exposto a seguir. Esse procedimento fundamenta-se na observação, decorrente da Proposição 2, de que cada vez que a regra da folga mínima for quebrada, quando da escolha da atividade a ser incorporada a um caminho em construção pelo processo *forward*

(ou *backward*), o caminho obtido terá duração menor que a duração do maior caminho, na medida exata da diferença entre a folga da atividade escolhida e a menor folga das atividades sucessoras (ou antecessoras).

## 6.2 Procedimento de busca de caminhos implícitos

**Passo 1:** Considere  $L[(P)]$  o conjunto formado inicialmente pelos  $k$  caminhos  $P^1, P^2, \dots, P^k$  identificados pelo procedimento de busca de caminhos explícitos, listados em ordem não crescente dos valores de  $T(P^j)$ ,  $j = 1, 2, \dots, k$ . Faça  $j = 1$ .

**Passo 2:** Retome a atividade semente  $(p, q)$  e o caminho parcial  $P^j(s/p)$  que geraram o caminho  $P^j$ . Faça  $i = 1$  e construa o caminho parcial  $P^j_i(q/t)$  aplicando o processo *forward* a partir do vértice  $q$ , rompendo, no vértice  $u$  mais próximo de  $q$ , a regra de escolha pela mínima folga, escolhendo, se existir, uma atividade  $(u, v_i)$  que ramifica para fora do caminho parcial  $P^j_i(q/t)$ , tal que  $\min\_folga(u) < f(u, v_i) \leq [T(P^j) - T(P^{j+n})]$ , em que  $P^{j+n}$  é o caminho mais próximo de  $P^j$  em  $L[(P)]$ , tal que  $T(P^{j+n}) < T(P^j)$ , e  $\min\_folga(u)$  é igual a folga da atividade de menor folga que emerge do vértice  $u$  (aquela atividade que foi escolhida na construção do caminho parcial  $P^j(p/t)$ ). Complemente o caminho  $P^j_i(q/t)$  aplicando o processo *forward* a partir do vértice  $v_i$ . Caso não exista a atividade  $(u, v_i)$ , siga para o Passo 4.

**Passo 3:** Se  $P^j_i = P^j(s/p) + (p, q) + P^j_i(q/t)$  for um novo caminho, insira-o em  $L[(P)]$ , observando seu posicionamento determinado por  $T(P^j_i)$ . Faça  $i = i + 1$  e aplique novamente o procedimento do Passo 2, impedindo a escolha das atividades  $(u, v_i)$ ,  $i = 1, 2, \dots, i-1$ .

**Passo 4:** Retome a atividade semente  $(p, q)$  e o caminho parcial  $P^j(q/t)$  que geraram o caminho  $P^j$ . Faça  $i = 1$  e construa o caminho parcial  $P^j_i(s/p)$  aplicando o processo *backward* a partir do vértice  $p$ , rompendo, no vértice  $v$  mais próximo de  $p$ , a regra de escolha pela mínima folga, escolhendo, se existir, uma atividade  $(u_i, v)$  que ramifica para fora do caminho parcial  $P^j_i(s/p)$ , tal que  $\min\_folga(v) < f(u_i, v) \leq [T(P^j) - T(P^{j+n})]$ . Caso não exista a atividade  $(u_i, v)$ , siga para o Passo 6.

**Passo 5:** Se  $P^j_i = P^j_i(s/p) + (p, q) + P^j(q/t)$  for um novo caminho, insira-o em  $L[(P)]$ , observando seu posicionamento determinado por  $T(P^j_i)$ . Faça  $i = i + 1$  e retorne ao Passo 4, impedindo a escolha das atividades  $(u_i, v)$ ,  $i = 1, 2, \dots, i - 1$ .

**Passo 6:** Elimine da lista  $L[(P)]$  o caminho  $P^j$ . Se  $[T(P^j) - T(P^{j+n})] = 0$ , pare. Caso contrário, faça  $j = j + 1$  e retorne ao Passo 1.

Cada caminho  $P^j$  identificado pelo procedimento de busca de caminhos explícitos pela folga pode gerar caminhos implícitos  $P^j_i$ , tais que  $T(P^j) > T(P^j_i) \geq T(P^{j+n})$ , o que explica o critério de escolha da atividade para ramificar

fora do caminho  $P^i$ , conforme indicado nos passos 2 e 4 do procedimento de busca de caminhos implícitos.

Aplicando-se esse procedimento ao exemplo da Figura 2, o conjunto  $L[(P)]$  é formado, inicialmente, por:  $P^1 = \{1, 3, 5, 6, 9\}$ , com duração igual a 13;  $P^2 = \{1, 2, 3, 5, 6, 9\}$ , com duração igual a 12;  $P^3 = \{1, 3, 6, 9\}$ , com duração também igual a 12; e  $P^4 = \{1, 3, 5, 6, 8, 9\}$ , com duração igual a 11. Fazendo-se  $j = 1$ , o próximo caminho  $P^{j+n}$  de  $L[(P)]$ , tal que  $T(P^{j+n}) < T(P^j)$ , é  $P^2$ . Assim,  $T(P^1) - T(P^2) = 1$ . Como a atividade semente de  $P^1$  é (1, 3), vamos, por meio do processo *forward* a partir do vértice 3, em busca de uma atividade  $(u, v_1)$  com folga igual a 1. Identifica-se  $(u, v_1) = (3, 6)$ , que origina o caminho parcial  $P^1_1(3/9) = (3, 6, 9)$ . Então,  $P^1_1 = \{1, 3, 6, 9\}$ , que já está presente em  $L[(P)]$ . Continuando-se a explorar o caminho  $P^1$ , ainda pelo processo *forward* a partir do vértice 3, verifica-se que todas as demais atividades que dele ramificam possuem folga superior a 1. Como o processo *backward* não é aplicável a partir do vértice 1, fazemos  $j = 2$  e encontramos  $P^{2+n} = P^4$ , com  $T(P^2) - T(P^4) = 1$ . A atividade semente de  $P^2$  é (2, 3). Aplicando-se o processo *forward* a partir do vértice 3, identifica-se  $(u, v_1) = (3, 6)$ , o que origina o caminho parcial  $P^2_1(3/t) = \{3, 6, 9\}$  e, como  $P^2(s/2) = \{1, 2\}$ , temos  $P^2_1 = \{1, 2, 3, 6, 9\}$ , com duração 11. Como se trata de um caminho novo, é inserido em  $L[(P)]$ , ocupando a quinta posição. Explorando-se ainda o caminho  $P^2$ , pelo processo *forward* verifica-se que todas as demais atividades que dele ramificam possuem folga superior a 1. Como, pelo processo *backward*, não é possível ramificar fora de  $P^2$  a partir do vértice 2, fazemos  $j = 3$  e encontramos  $P^{3+n} = P^4$  e  $T(P^3) - T(P^4) = 1$ . A atividade semente de  $P^3$  é (3, 6). Verifica-se que não existe atividade  $(u, v)$  que possa ramificar fora do caminho  $P^3$ , quer seja pelo processo *forward* ou *backward*. Fazendo-se  $j = 4$ , temos  $P^{4+n} = P^5$  e  $T(P^4) - T(P^5) = 0$ . Encerra-se, portanto, o processo de busca.

O procedimento de busca de caminhos implícitos e o algoritmo de Ju e Saleh (1991) exigem esforço computacional parecido. Porém, o procedimento aqui proposto trabalha com folgas de atividades, que são mais fáceis de determinar do que as folgas relativas, conforme proposto em Ju e Saleh (1991).

A seguir, são apresentados os ensaios computacionais realizados com o objetivo de avaliar a frequência com que os caminhos não explícitos aparecem em redes PERT.

## 7 Ensaios Computacionais

Foram realizados ensaios computacionais para verificar a eficácia (nível de acerto) do procedimento de busca de caminhos explícitos pela folga na identificação de todos os caminhos do projeto com duração maior que um dado valor  $T^*$ . Para isto, os algoritmos propostos foram codificados na linguagem C e executados em um

microcomputador com processador Intel Core 2 Duo de 2 GHz e 2 GB de RAM.

Nos ensaios conduzidos, foram consideradas duas situações normalmente presentes no ambiente que envolve projetos que se realizam uma única vez:

- projetos com diversos níveis de complexidade; e
- projetos com diferentes níveis de incerteza.

O nível de complexidade do projeto foi definido pela topologia da rede (número de nós e número de arcos) que o representa. O número de arcos foi determinado por meio da densidade,  $d$ , da rede, a qual é expressa em função de  $n$ , número de nós. A densidade da rede é dada pelo quociente entre o número de arcos da rede e o número máximo possível de arcos numa rede PERT, o qual é dado pela expressão  $[\sum_{i=1}^n (n-i) = n(n-1)/2]$ .

Foram ensaiadas redes com topologia ( $n = 10, d = 0,6$ ), ( $n = 20, d = 0,4$ ) e ( $n = 30, d = 0,3$ ), que determinam, nesta ordem, três níveis crescentes de complexidade do projeto.

O nível de incerteza do projeto foi definido pela diferença entre a duração otimista,  $to(i, j)$ , e pessimista,  $tp(i, j)$ , das atividades  $(i, j)$  do projeto. Considerou-se que quanto maior essa diferença, maior a incerteza sobre a duração das atividades e, portanto, sobre o projeto. Foram ensaiadas redes com três níveis de incerteza, determinados pelos seguintes valores de  $\delta$ , em que  $\delta$  é o parâmetro da expressão  $tp(i, j) = (1 + \delta).to(i, j)$ :

- nível 1 de incerteza:  $\delta \in [0,2; 0,5]$ ;
- nível 2 de incerteza:  $\delta \in [0,4; 0,8]$ ; e
- nível 3 de incerteza:  $\delta \in [0,5; 1,0]$ .

Criaram-se, assim, nove redes topologicamente distintas, considerando-se as combinações possíveis entre os níveis de complexidade e os níveis de incerteza adotados.

A duração das atividades foi considerada uma variável aleatória com distribuição beta de probabilidade, que é uma das que melhor representam a variável aleatória duração de uma atividade que se realiza uma única vez. A Equação 1 apresenta a função densidade de probabilidades para a variável aleatória  $x \in [0, 1]$  com distribuição beta de probabilidades. Seus parâmetros,  $\alpha$  e  $\beta$ , são responsáveis pela forma que a função assume. Por meio da uma transformação linear, expressa pela Equação 2, pode-se definir o domínio da variável aleatória  $T$ , duração da atividade, no intervalo  $(to, tp)$ .

$$f(x) = \frac{x^\alpha (1-x)^\beta}{\int_0^1 x^\alpha (1-x)^\beta dx}, 0 < x < 1, \alpha > -1, \beta > -1 \quad (1)$$

$$T = x.(tp - to) + to \quad (2)$$

A configuração da rede é definida por sua topologia, definida pelo conjunto de nós e arcos, e pela duração de

cada atividade (arco), definida pela distribuição de probabilidade que a caracteriza.

A topologia de cada uma das nove redes ensaiadas foi determinada pela escolha aleatória de arcos, de forma a satisfazer o par  $(n, d)$ . O comportamento da duração de cada atividade  $(i, j)$  foi estabelecido determinando-se, aleatoriamente, os seguintes parâmetros:

- f)  $to(i, j) \in [10, 30]$ , uniformemente distribuído;
- g)  $\delta$ , uniformemente distribuído nos intervalos definidos anteriormente, conforme o nível de incerteza sobre a duração das atividades;
- h)  $tp(i, j) = (1 + \delta) \cdot to(i, j)$ ; e
- i)  $\alpha$  e  $\beta$ , uniformemente distribuídos no intervalo  $[2, 6]$ .

Gerada uma rede, foram identificados, por meio de enumeração completa, todos os seus caminhos e determinadas as durações esperada ( $Te$ ) e acelerada ( $Ta$ ) do projeto. A duração esperada do projeto é igual a duração

do seu caminho mais longo quando todas as suas atividades  $(i, j)$  estão com duração média,  $\bar{t}(i, j)$ , e a duração acelerada é obtida do caminho mais longo quando as atividades estão com duração otimista,  $to(i, j)$ .

Os valores de  $T^*$  foram definidos por meio da expressão  $T^* = Te - b(Te - Ta)$ , com  $b = \{0,1; 0,2; \dots; 1,0\}$ . A eficácia do procedimento proposto foi avaliada comparando-se  $Np$  com  $Nv$ , em que  $Np$  é o número de caminhos com duração maior que  $T^*$ , identificado pelo procedimento proposto, e  $Nv$  é o verdadeiro número de caminhos do projeto com duração maior que  $T^*$ , identificado por enumeração completa. Os testes foram realizados para as atividades do projeto com duração igual ao seu tempo esperado.

O Quadro 2 exibe o resultado dos ensaios. O número de caminhos identificados pela enumeração completa está reproduzido na coluna  $Nv$  e o número de caminhos identificados pelo procedimento de busca de caminhos

**Quadro 2.** Resultados dos ensaios para o algoritmo da folga mínima.

Nível de Incerteza	b	Topologia da rede								
		n = 10, d = 0,60			n = 20, d = 0,40			n = 30, d = 0,30		
		Nv	Np	Efic	Nv	Np	Efic	Nv	Np	Efic
$\delta \in [0,2; 0,5]$	0,1	1	1	1,00	2	2	1,00	1	1	1,00
	0,2	2	2	1,00	2	2	1,00	3	3	1,00
	0,3	2	2	1,00	2	2	1,00	6	5	0,83
	0,4	2	2	1,00	2	2	1,00	13	10	0,70
	0,5	2	2	1,00	5	4	0,80	21	12	0,57
	0,6	2	2	1,00	7	5	0,71	31	13	0,42
	0,7	2	2	1,00	8	6	0,75	44	15	0,34
	0,8	3	3	1,00	10	7	0,70	61	18	0,30
	0,9	4	4	1,00	13	10	0,77	89	20	0,22
	1,0	4	4	1,00	17	10	0,59	117	23	0,20
$\delta \in [0,4; 0,8]$	0,1	1	1	1,00	2	2	1,00	1	1	1,00
	0,2	2	2	1,00	2	2	1,00	3	3	1,00
	0,3	2	2	1,00	4	4	1,00	6	5	0,83
	0,4	3	3	1,00	13	9	0,69	10	7	0,70
	0,5	4	4	1,00	2	12	0,6	14	10	0,71
	0,6	5	5	1,00	32	15	0,47	26	18	0,69
	0,7	6	5	0,83	57	19	0,33	39	19	0,49
	0,8	9	6	0,67	78	21	0,37	62	21	0,34
	0,9	9	6	0,67	112	23	0,21	79	22	0,28
	1,0	11	6	0,55	157	24	0,15	109	27	0,25
$\delta \in [0,5; 1,0]$	0,1	1	1	1,00	3	3	1,00	2	2	1,00
	0,2	1	1	1,00	3	3	1,00	4	3	0,75
	0,3	1	1	1,00	4	4	1,00	4	3	0,75
	0,4	2	2	1,00	8	8	1,00	13	8	0,62
	0,5	3	3	1,00	15	12	0,80	26	13	0,50
	0,6	4	4	1,00	18	13	0,72	37	17	0,46
	0,7	5	5	1,00	24	14	0,58	53	2	0,38
	0,8	6	6	1,00	34	17	0,50	84	23	0,27
	0,9	10	9	0,90	49	2	0,41	133	28	0,21
	1,0	12	9	0,75	59	2	0,34	183	33	0,18

explícitos pela folga está reproduzido na coluna  $N_p$ . A coluna Efic fornece a eficácia do procedimento proposto, dada pela expressão  $Efic = N_p/N_v$ .

Verifica-se que a eficácia do procedimento proposto se reduz progressivamente com o aumento da complexidade do projeto (topologia da rede, dada pelo par  $[n, d]$ ) e com o aumento do nível de incerteza do projeto (dado pelo intervalo de variação do parâmetro  $\delta$ ), podendo funcionar bem quando se buscam pequenas reduções na duração do projeto (até 30% da máxima redução possível, conforme os ensaios realizados).

Também por meio de testes computacionais foi verificado o comportamento do procedimento de busca de caminhos explícitos pela folga, quando aplicado a redes probabilísticas (os resultados estão expostos no Quadro 3). Nesses testes, cada uma das nove redes definidas anteriormente foi ensaiada 10000 vezes. Em cada ensaio a duração das atividades,  $t(i, j)$ , assumia um valor entre suas durações otimista e pessimista, identificado aleatoriamente. Para tanto, obteve-se, pelo método de Monte Carlo, os valores da variável  $x(i, j)$ , para cada atividade  $(i, j)$  da rede, utilizando-se a Equação 1, e o correspondente valor da variável  $t(i, j)$ , utilizando-se a Equação 2. Esse processo corresponde a “realizar” o projeto 10000 vezes. O “verdadeiro” caminho  $k$ -crítico do projeto,  $P^k$  (aquele que domina em probabilidade os demais caminhos  $P^i, j > k$ ), foi considerado aquele caminho que apareceu com maior frequência na  $k$ -ésima posição.

No Quadro 3, os caminhos estão representados por números a eles atribuídos quando da geração da rede aleatória. A coluna  $P_p^k$  do Quadro 3 apresenta os caminhos  $k$ -críticos localizados pelo procedimento de busca de caminhos explícitos pela folga, e a coluna  $P_v^k$  apresenta o verdadeiro caminho  $k$ -crítico. Os caminhos estão listados nas linhas das células do Quadro 3 conforme a ordem crescente de  $k$ . Assim, para a rede  $n = 30, d = 0,30$  e  $\delta \in [0,2; 0,5]$ , o primeiro e o segundo caminhos mais longos do projeto localizados pelo algoritmo da folga mínima são os de números 68 e 788, respectivamente, enquanto que o primeiro e o segundo verdadeiros caminhos mais longos são os de números 68 e 83, respectivamente. Com o objetivo de verificar se um possível erro em detectar o verdadeiro caminho  $k$ -crítico é real ou aparente, o Quadro 3 apresenta, nas colunas identificadas por  $IC[\bar{T}]$ , os intervalos de 95% de confiança para a duração média dos caminhos,  $\bar{T}$  (os extremos do intervalo foram aproximados para valores inteiros). Com isso, verifica-se que, na verdade, não ocorreu erro na identificação dos dois caminhos mais longos para a rede  $n = 30, d = 0,30$  e  $\delta \in [0,2; 0,5]$ .

## 8 Conclusões

O algoritmo de Dodin (1984) não se apresenta como uma boa alternativa para a busca dos caminhos mais longos de um projeto. O fato de trabalhar com redes

**Quadro 3.** Resultados dos ensaios do Algoritmo da folga mínima em redes probabilísticas.

Nível de incerteza	Topologia da rede								
	n = 10, d = 0,60			n = 20, d = 0,40			n = 30, d = 0,30		
	$P_p^k$	$P_v^k$	$IC[\bar{T}]$	$P_A^k$	$P_V^k$	$IC[\bar{T}]$	$P_A^k$	$P_V^k$	$IC[\bar{T}]$
$\delta \in [0,2; 0,5]$	1	1	162-174	227	227	223-238	68	68	306-324
	10	10	157-171	66	66	221-236	788		300-317
	5	5	144-156	228	228	206-222	83	83	300-317
	2	2	142-153	232		206-221		788	300-317
	3	3	132-149		67	204-220	2371	2371	297-313
					232	206-221	4494	4494	294-311
$\delta \in [0,4; 0,8]$				132		202-217			
	6	6	201-227	80	80	357-393	259	259	358-389
	1	1	193-217	479	479	351-386	271	271	345-378
	9	9	184-208	316	316	339-373	197	197	345-378
	36	36	180-204	56	56	337-368	239	239	340-372
	7	7	173-198	181		331-364	1145	1145	338-368
$\delta \in [0,5; 1,0]$					455	331-361			
	11	11	176-203	69	69	268-301	343	343	408-449
	20	20	156-181	1	1	265-299	294	294	406-447
	25	25	153-178	214	214	262-296	344	344	389-430
	12	12	151-175	46	46	250-282		295	387-427
	16	16	145-171	101	101	243-275	352	352	374-413
						2003		372-410	

probabilísticas não lhe adiciona vantagem, pois os demais algoritmos apresentados podem ser aplicados também em redes desta natureza.

Embora não se possa comparar de forma adequada a eficácia do algoritmo de Dodin (1984) com o procedimento de busca de caminhos explícitos pela folga, uma vez que os ensaios não foram realizados sobre o mesmo conjunto de redes, a comparação dos resultados expressos nos Quadros 1 e 3 sugere que este último procedimento possui melhor desempenho do que o primeiro. A maior rede ensaiada por Dodin possui 30 nós e 90 arcos, enquanto a maior rede ensaiada neste trabalho possui 30 nós e 130 arcos.

O algoritmo de Ju e Saleh (1991) requer menor esforço computacional do que aquele apresentado por Yen et al. (1989), conforme demonstraram os primeiros autores. Comparando-se o procedimento de busca de caminhos explícitos pela folga com o algoritmo de Ju e Saleh (1991), desconsiderando-se o fato de ser mais simples determinar a folga de uma atividade do que sua folga relativa, verifica-se que os esforços computacionais para a preparação da rede (que corresponde aos passos 1 e 2 do primeiro e aos passos de 1 a 4 do segundo algoritmo) se equivalem (em ambos os algoritmos cada nó da rede

deve ser rotulado e para cada atividade uma folga deve ser determinada), assim como o processo de identificação de cada novo caminho (passos 5, 7 e 8 do primeiro e passos 5, 6 e 7 do segundo algoritmo). O passo 5 do algoritmo de Ju e Saleh (1991) é o que faz a diferença, imprimindo a esse algoritmo um nível de complexidade superior ao do procedimento de busca de caminhos explícitos pela folga. Essa conclusão também é válida com respeito ao procedimento de busca de caminhos implícitos, uma vez que sua aplicação não requer nenhuma outra informação além daquelas contidas na rede original, e a rotina para construção de cada novo caminho é a mesma daquela adotada pelo procedimento de busca de caminhos explícitos pela folga, ou seja, a escolha de uma nova atividade para compor o caminho é feita entre aquelas atividades sucessoras imediatas à última escolhida, conforme determinam os passos 2 e 4 do procedimento de busca de caminhos implícitos.

Quanto à expectativa de eficácia do algoritmo de busca de caminhos explícitos, o que lhe confere grande vantagem computacional em relação aos demais algoritmos existentes na literatura, devido à sua simplicidade, verifica-se que ela se concretiza quando se buscam pequenas reduções na duração do projeto.

## Determination of K most critical paths in PERT networks

### Abstract

*This paper presents a study about the most important methods to determine the K most critical paths in PERT networks. When  $k = 1$ , it means the longest path, traditionally known as the critical path; when  $k = 2$ , it means the second longest path, and so on. Three algorithms available in the literature are discussed, and a new algorithm, known as minimum slack algorithm and which presents some advantages over the others, is proposed. The performance of the minimum slack algorithm when applied to PERT networks is evaluated using a simulation process.*

**Keywords:** Project management. CPM/PERT. K most critical paths.

### Referências Bibliográficas

- BORNA, A.; PROGLER, C.; BLAAUW, D. Correlation analysis of CD-variation and circuit performance under multiple sources of variability. In: Liebmann, L.W. (Ed.) **Design and Process Integration for Microelectronic Manufacturing III** (Proceedings of SPIE). San Jose, CA, v. 5756, p. 168-177, may 2005, ISBN: 0819457361.
- CONTADOR, J. L. Algoritmo da folga mínima para determinação do caminho k-crítico em redes PERT. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 33, 2001, Campos do Jordão, SP. **Anais...** Rio de Janeiro: SOBRAPO, p. 1595-1602.
- DODIN, B. Determining the k Most Critical Paths in PERT Networks. **Operations Research**, v. 32, n. 4, p. 859-877, jul-aug, 1984.
- GHOMI, S. M. T. F.; RABBANI, M. Approximating completion time distribution function for stochastic PERT networks. **Iranian Journal of Science and Technology**, v. 26, n. B1, p. 107-110, 2001.

- JU, Y. C.; SALEH, A. R. Incremental techniques for the identification of statically sensitizable critical paths. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 28, 1991, San Francisco, CA. **Proceedings...** New York, NY: ACM, p. 541-546, ISBN: 0-89791-395-7.
- KIM, S. J.; BOYD, S. P.; YUN, S.; PATIL, D. D.; HOROWITZ, M.A. A heuristic for optimizing stochastic activity networks with applications to statistical digital circuit sizing. **Optimization and Engineering**, v. 8, n. 4, p. 397-430, dec. 2007.
- LUO, T.; NEWMARK, D.; PAN, D.Z. A new LP based incremental timing driven placement for high performance designs. In: ANNUAL CONFERENCE ON DESIGN AUTOMATION, 43., 2006, San Francisco, CA. **Proceedings...** New York, NY: ACM, p. 1115-1120, ISBN:1-59593-381-6.
- MODER, J. J.; PHILLIPS, C. R.; DAVIS, E. W. **Project Management with CPM, PERT and Precedence Diagramming**. 3. ed., New York: Van Nostrand Reinhold, 1983. 389 p.
- SOROUSH, H.M. The most critical path in a PERT network. **The Journal of Operations Research Society**, v. 45, n. 3, p. 287-300, mar. 1994.
- YEN, S. H.; DU, D. H.; GHANTA, S. Efficient algorithm for extracting the k most critical paths in timing analysis. In: ACM/IEEE Design Automation Conference, 26, 1989, Las Vegas, NV. **Proceedings...** New York, NY: ACM, p. 649-654, ISBN: 0-89791-310-8.

---

### **Sobre os autores**

---

#### **José Luiz Contador**

Programa de Mestrado em Administração das Micro e Pequenas Empresas,  
Faculdade Campo Limpo Paulista – FACCAMP,  
Rua Guatemala, 167, Jardim América, CEP 13231-230, Campo Limpo Paulista, SP, Brasil,  
e-mail: jluiuz@feg.unesp.br

#### **Edson Luiz França Senne**

Faculdade de Engenharia, Campus de Guaratinguetá, Universidade Estadual Paulista,  
Av. Ariberto Pereira da Cunha, 333, CEP 12516-410, Guaratinguetá, SP, Brasil,  
e-mail: elfsenne@feg.unesp.br

**Agradecimentos:** Os autores agradecem a três revisores anônimos pelos valiosos comentários e sugestões.

Recebido em 23/1/07  
Aceito em 16/11/07