Acelerando o método ECM através da computação paralela

Adriana B. de Paula Molgora, **Fabrício S. de Paula**
Curso de Ciência da Computação, UEMS,
79804-970, Dourados, MS
E-mail: abmol@terra.com.br, fabricio@uems.br

Marcelo O. Dias
Instituto de Computação, UNICAMP,
13083-970, Campinas, SP
E-mail: maruero@gmail.com.

Resumo: *O método ECM (Elliptic Curve Method) é um dos principais métodos de fatoração de números inteiros. Este método explora as propriedades algébricas dos pontos de uma curva elítica para encontrar um fator de um número. O principal objetivo deste trabalho é apresentar o ganho de desempenho obtido com a utilização do processamento paralelo na implementação desse método de fatoração.*

1 Introdução

Criptografia é a ciência que estuda métodos para codificar e decodificar mensagens. O uso desta é imprescindível quando se deseja privacidade e segurança na troca de informações [3]. Um dos sistemas criptográficos mais utilizados atualmente é o RSA , cuja segurança baseia-se na dificuldade de fatorar números grandes [16, 17].

O problema de fatoração de números inteiros está entre os mais antigos problemas matemáticos abordados pela humanidade e, até o momento, não se conhece nenhum método que realize a fatoração de qualquer número inteiro composto em tempo satisfatório. Isso, tem motivado diversos estudos através dos quais foram desenvolvidos alguns métodos de fatoração [14, 9, 15, 10] e, dentre esses, o método ECM, que explora as propriedades das curvas elíticas para encontrar um fator de um número inteiro composto.

O tempo de execução do método ECM depende diretamente do tamanho do fator a ser encontrado, ou seja, quanto maior o fator, maior será o tempo gasto na fatoração. O sucesso na fatoração também depende de escolhas aleatórias de curvas elíticas e, caso a fatoração não seja bem sucedida utilizando-se uma determinada curva , é necessário realizar outras tentativas de fatoração a partir de outras curvas elíticas. Nesse sentido, uma maneira de agilizar o processo de fatoração seria a utilização da computação paralela, através da qual pode-se realizar várias escolhas de curvas ao mesmo tempo e, consequentemente, várias tentativas de fatoração.

Este artigo apresenta um estudo comparativo do método ECM em relação à duas implementações do mesmo, sendo uma seqüencial e outra paralela. O texto está organizado como segue. A Seção 2 descreve processo de fatoração realizado pelo método ECM e a Seção 3 apresenta alguns conceitos básicos de algoritmos paralelos. A Seção 4 apresenta os aspectos das implementações realizadas e os testes efetuados. Finalmente, na Seção 5 são apresentadas as considerações finais desse trabalho.

2 ECM

O método ECM, foi descoberto por H.W. Lenstra Jr., em 1985 [9]. Para um melhor entendimento do mesmo, a seguir são apresentados, de forma sucinta, alguns conceitos elementares da teoria de curvas elíticas baseados em [2, 6, 18, 19].

2.1 Curvas elíticas

Em coordenadas afins, uma curva elítica definida sobre um corpo de caracterrística $\neq 2, 3$

pode ser escrita na forma normal de Weierstrass por:

$$y^2 = f(x) = x^3 + ax + b$$

onde

$$4a^3 + 27b^2 \neq 0$$

o que garante que f não tem raízes múltiplas.

Sobre uma curva elítica pode-se definir uma operação de adição de pontos pertencentes à curva, possuindo estrutura de grupo abeliano.

Operação de adição: Seja O um ponto qualquer sobre C. Considere a operação $+$ que a cada par (P, Q) de pontos de C associa o ponto $P + Q$ sobre C definido por $P + Q = O * (P * Q)$. Assim, dados dois pontos P e Q, encontra-se o terceiro ponto $P * Q$ traçando a reta L_1 que passa por P e Q; em seguida traça-se a reta L_2 que passa por O e $P*Q$ e encontra-se o terceiro ponto que é $P+Q$. Ver Figura 1.

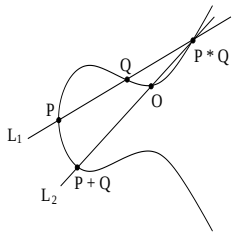

Figura 1: Obtenção do ponto $P + Q$

Considerando-se o ponto O como sendo o ponto no infinito (o ponto onde as retas verticais se intersectam), para se adicionar dois pontos P_1 e P_2 traçamos primeiramente a reta que passa por P_1 e P_2 encontrando o ponto de intersecção entre a reta e a curva. Em seguida traçamos a reta que passa por O e por $P_1 * P_2$ que é exatamente a reta vertical que passa por $P_1 * P_2$. O ponto $P_1 + P_2$ será o ponto simétrico de $P_1 * P_2$. Ver Figura 2.

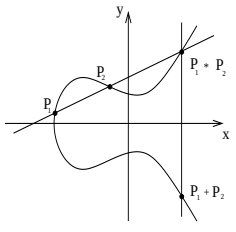

Figura 2: Obtenção do ponto $P + Q$ considerando o ponto O como sendo o ponto no infinito

A multiplicação de um ponto P de uma curva C por um inteiro $k > 2$ denominada **operação escalar de um ponto** é a principal operação realizada no processo de fatoração pelo método ECM. Essa operação pode ser efetuada utilizando-se as fórmulas de adição e duplicação de ponto, ou seja, adicionando o ponto a ele mesmo tantas vezes quanto for o fator de multiplicação:

$$kP = \underbrace{P + P + \cdots + P}_{k\ vezes}$$

2.2 Fórmulas algébricas para a adição e duplicação de pontos

Algebricamente, a operação de adição de pontos da curva pode ser realizada utilizando-se coordenadas afins ou projetivas. Em coordenadas projetivas, a curva pode ser representada na forma de Montgomery [12] por $by^2z = x^3 + ax^2z + xz^2$, com $b(a^2 - 4) \neq 0$. Assim, $(x : y : z)$ com $z \neq 0$ representa o ponto $(y/z, x/z)$ em coordenadas afins. Se $z = 0$, então tem-se o ponto no infinito O representado por $(0 : 1 : 0)$ em coordenadas projetivas.

Suyama em [20], sugere uma parametrização da curva elítica na forma de Montgomery, que possibilita a obtenção de curvas cuja ordem $\#C(\mathbb{Z}_p)$ seja divisível por 12. Para isso é escolhido um inteiro $\sigma > 5$ e são calculados $u = \sigma^2 - 5$ e $v = 4\sigma$. Então são definidos os parâmetros a e b da curva por

$$a_\sigma = \frac{(v-u)^3(3u+v)}{4u^3v} - 2,\ b_\sigma = \frac{u}{v^3}$$

e o ponto inicial $P_\sigma(x_\sigma, y_\sigma, z_\sigma)$ por

$$x_\sigma = u^3,\ y_\sigma = (\sigma^2 - 1)(\sigma^2 - 25)(\sigma^4 - 25),\ z_\sigma = v^3$$

Pode-se notar que a obtenção de um ponto inicial $P_0 = (x_0, y_0, z_0)$ da curva, depende apenas do parâmetro σ.

No método ECM, pode-se calcular a adição de pontos da curva utilizando-se apenas as coordenadas x e z do ponto [12]. Uma maneira de otimizar esse processo, é escolher a coordenada z_0 do ponto inicial P_0 igual 1. Pela parametrização de Suyama, tomando-se $z_0 \equiv z_\sigma/z_\sigma \equiv 1 \pmod n$, tem-se $x_0 \equiv x_\sigma/z_\sigma \equiv (4u^4 \cdot \frac{1}{4u^3v})^3 \pmod n$.

Considerando-se os pontos $P(x_P : z_P)$ e $Q(x_Q : z_Q)$ com $x_P x_Q(x_P - x_Q) \neq 0$, $P_0 = (x_0 : z_0) = (x_{P-Q} : z_{P-Q}) = P - Q$, e $a_{24} = (a + 2)/4$, onde a é o parâmetro da curva, tem-se a adição de pontos $P + Q = (x_{P+Q} : z_{P+Q})$ onde:

$$x_{P+Q} = z_{P-Q}[(x_P - z_P)(x_Q + z_Q) + (x_P + z_P)(x_Q - z_Q)]^2$$

e

$$z_{P+Q} = x_{P-Q}[(x_P - z_P)(x_Q + z_Q) - (x_P + z_P)(x_Q - z_Q)]^2$$

A duplicação do ponto P dada por $2P = (x_{2P} : z_{2P})$ pode ser obtida calculando-se

$$4x_P z_P = (x_P + z_P)^2 - (x_P - z_P)^2$$
$$x_{2P} = (x_P + z_P)^2 (x_P - z_P)^2$$
$$z_{2P} = (4x_P z_P) - (x_P - z_P)^2 + a_2 4(4x_P z_P)$$

2.3 Fatoração de inteiros utilizando o método ECM

A idéia do método ECM é baseada no método $p-1$ de Pollard [14]. Para encontrar um fator p de n, supõe-se um inteiro k tal que $\#C(\mathbb{Z}_p)|k$. Segue do Teorema de Lagrange [6] que a ordem de qualquer ponto de $C(\mathbb{Z}_p)$ divide k. Em particular $kP = O$. Sendo $kP = (x : y : z)$ onde $(x : y : z)$ é uma terna de coordenadas normalizadas, então $z \equiv 0 \bmod p$, logo $p|z$. Assim p é fator comum de n e z, donde segue que $p|\text{mdc}(z, n)$.

Na prática, para para encontrar um fator p de n é necessário escolher um número k esperando-se que $\#C(\mathbb{Z}_p)|k$. Para isso, k é escolhido como um produto de primos pequenos elevados a potências adequadas, onde a expressão potência adequadas pode ser interpretada como potências pequenas. Em geral, dado um inteiro B, $k = \prod_{d_i^{a_i} \leq B} d_i^{a_i}$ onde os d_i's são números primos.

Depois de determinado o número k, escolhe-se um valor aleatório para σ para calcular as coordenadas do ponto inicial $P(x : y : z)$. Se for tomado $z = 1$, como proposto na Seção 2.2, basta calcular a coordenada x, pois a coordenada y não é necessária na operação de adição de pontos.

Depois da obtenção do ponto inicial, são utilizadas as fórmulas de adição e duplicação de pontos apresentadas na Seção 2.2 para calcular kP. As operações são realizadas módulo n. Se $\#C(\mathbb{Z}_p)|k$ então $kP = O$, logo $z \equiv 0 \bmod p$ e um fator desconhecido de n pode ser descoberto calculando-se o $\text{mdc}(z, n)$. Se $\text{mdc}(z, n) = d < n$ então d é fator não trivial de n. Caso contrário, é escolhido um novo valor para σ, calculado um novo ponto e repetido todo o processo.

Pode-se afirmar que o tempo esperado para o método ECM encontrar o menor fator p de n é de aproximadamente $O(exp(\sqrt{2 \ln p \ln \ln p}))$ operações de grupo [4]. Portanto, o método ECM apresenta um tempo de execução subexponencial. O pior caso do algoritmo ocorre quando $p \approx \sqrt{n}$.

3 Algoritmos paralelos

Enquanto em um algoritmo sequencial várias tarefas são executadas uma após a outra, em algoritmos paralelos várias tarefas podem ser executadas por diversos processadores simultaneamente [8, 1].

Para a verificar a qualidade de um algoritmo paralelo duas medidas utilizadas são o *speedup* e a eficiência. O *speedup* pode ser definido como a relação entre o tempo gasto para executar uma tarefa com 1 processador e o tempo gasto com p processadores, ou seja, *speedup* é a medida do ganho em tempo dada por $S_\mathtt{p} = T_s/T_\mathtt{p}$ onde T_s é o tempo gasto com 1 processador e $T_\mathtt{p}$ é o tempo gasto com p processadores.

O valor ideal para o *speedup* seria p. Porém, alguns fatores como a sobrecarga da comunicação entre os processadores, partes do código executável estritamente seqüenciais e o nível de paralelismo utilizado podem influenciar esse resultado. Em alguns casos o *speedup* obtido pode ser maior que o previsto, ou seja, $S_\mathtt{p} > \mathtt{p}$. Nesse caso tem-se um *speedup* superlinear, que pode ser causado pelas características do algoritmo paralelo ou pelas características da plataforma de hardware.

A eficiência trata da relação entre o *speedup* e o número de processadores sendo calculada por $e = S_\mathtt{p}/\mathtt{p}$. No caso em que *speedup* = p, a eficiência seria máxima e teria valor 1.

4 Implementação e testes

4.1 Ambiente de desenvolvimento

Foram realizadas duas implementações do método ECM sendo uma sequencial e outra paralela, sendo que a sequencial refere-se à implementação realizada em [5], onde o algoritmo foi implementado na linguagem C em uma plataforma Pentium D 3,4 GHz, 1024 MB de RAM, usando o sistema operacional Slackware 11.0.0.

Para a implementação paralela foi desen-

volvido um algoritmo paralelo BSP/CGM[1] [1] do método ECM e, como mecanismo de troca de mensagens, utilizou-se a biblioteca LAM/MPI [13]. Essa implementação foi construída utilizando-se a linguagem C sobre o sistema operacional Slackware 11.0.0. Em ambas implementações foram utilizadas para a representação de números de precisão arbitrária a biblioteca GMP [7]. O cálculo de k foi obtido através do parâmetro $B = \lfloor \sqrt[8]{n} \rfloor$ proposto em [11].

A Figura 3 apresenta um esboço do algoritmo paralelo que foi implementado. Os passos 1, 2 e 3 compõem a chamada parte seqüencial do algoritmo sendo que a parte paralela começa a partir do passo 4 sendo responsável por divulgar o número n a ser fatorado e o valor do inteiro k, calculado no passo 3.

Nos passos 5, 6 e 7 são gerados o ponto inicial, a curva elítica e realizado o cálculo de kP a partir do qual verifica-se se n foi fatorado. Em posse do n e do k todos os p processadores executam esses passos paralelamente, utilizando diferentes curvas e diferentes pontos iniciais. Isso é garantido dividindo-se, no passo 5, o intervalo de aleatoriedade do valor de λ igualmente entre os processadores.

O passo 8 é o responsável pela divulgação dos kP's, calculados e verificados no passo anterior pelos p processadores, e pela sincronização dos mesmos. É nele que é realizada a maioria da comunicação do algoritmo. Através dele é verificado se algum dos p processadores encontrou um fator não trivial de n. Em caso afirmativo, a execução do algoritmo paralelo pode ser finalizada e a saída é retornada ao usuário. Caso contrário, todos os p processadores devem retroceder ao passo 5 para a escolha de uma nova curva e um novo ponto inicial para uma nova tentativa de fatoração.

As duas implementações foram testadas utilizando números com dois fatores de 1 a 30 dígitos decimais onde cada caso de teste foi repetido 20 vezes e calculada a média aritmética dos resultados.

Em um primeiro momento, os testes foram

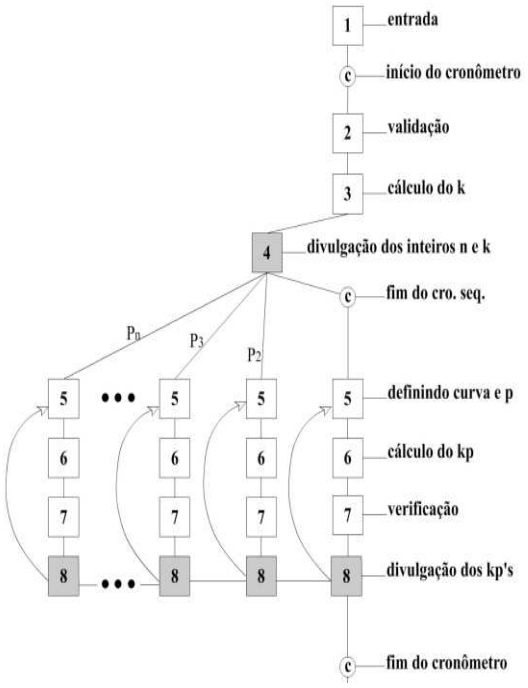

Figura 3: Algoritmo ECM paralelo

realizados em uma maquina multi-processada equipada com processador Xeon Dual Core[2] 3.4Ghz e com 2GB de memória de 400Mhz. Para medir a evolução da eficiência em comparação com o número de processadores, os testes com a implementação paralela foram repetidos para 2 e 4 processadores.

Num segundo momento, foram realizados testes com a implementação paralela do método utilizando 32 máquinas equipadas com processadores Pentium 4 de 3.2 Ghz e com 1GB de memória de 400Mhzs.

Foram obtidos resultados sobre o tempo de execução gasto no processo de fatoração nas implementações sequencial e paralela do método ECM. Além disso, foram calculados o *speedup* e a eficiência dessas implementações.

4.2 Análise dos resultados obtidos

A Tabela 1 mostra a média dos tempos de execução, em segundos, obtida com as implementações seqüencial e paralela do método ECM. A primeira coluna apresenta o número de dígitos decimais do fator de n. Nas demais colunas são apresentados os tempos de execução para

[1] Num algoritmo BSP/CGM a comunicação global é claramente separada da comunicação local. Realiza-se uma distribuição do problema pelos p processadores, e cada processador começa a executar uma rodada de computação alternando com uma rodada de comunicação global entre os p processadores, até que o problema seja resolvido.

[2] A tecnologia Dual Core traz dois processadores inteiros dentro de um mesmo invólucro. Equipado com a tecnologia *HiperThreading* – que simula a existência de dois processadores em cada núcleo – o sistema operacional reconhece um processador Xeon Dual Core como sendo quatro processadores.

as implementações sequencial, paralela com 2 processadores e paralela com 4 processadores, respectivamente.

dígitos	Seq.	2 proc.	4 proc.
1	0,000026	0,000282	0,000342
2	0,000034	0,000149	0,000444
3	0,000073	0,000193	0,000401
4	0,000367	0,000456	0,000761
5	0,003787	0,002107	0,003078
6	0,005339	0,002967	0,003281
7	0,008670	0,008702	0,006193
8	0,020742	0,014453	0,020583
9	0,052164	0,023306	0,032439
10	0,070426	0,066533	0,060619
11	0,111741	0,070113	0,112461
12	0,455442	0,189606	0,281586
13	0,616726	0,355734	0,328903
14	1,475478	0,838865	0,907607
15	2,184925	1,229923	1,155833
16	5,290828	4,985186	3,330895
17	10,597437	6,410346	5,939737
18	26,628722	11,242393	11,300536
19	55,132109	25,425687	23,379184
20	98,740049	38,598526	51,575918
21	144,818884	54,068062	74,314128
22	216,960636	174,5062630	108,338596
23	474,088544	250,432624	306,963545
24	614,346255	568,022386	810,542629
25	1155,149328	797,762854	816,940995
26	3806,466526	2203,891385	2311,437352
27	4383,764706	4251,222164	3153,801003
28	7352,177194	6350,165322	7208,516139
29	22194,200180	15494,653490	13569,087260
30	63906,148132	22502,623020	19890,356680

Tabela 1: Tempo de Execução

Os resultados mostrados na Tabela 1 também podem ser visualizados nas figuras 4 e 5. Na Figura 4 são apresentados os resultados obtidos nos testes de números com fatores de 1 a 20 dígitos decimais. A Figura 5 completa a apresentação com os resultados obtidos nos testes com números compostos de fatores de 21 a 30 dígitos. Nelas pode-se notar que houve realmente um ganho de desempenho utilizando a implementação paralela.

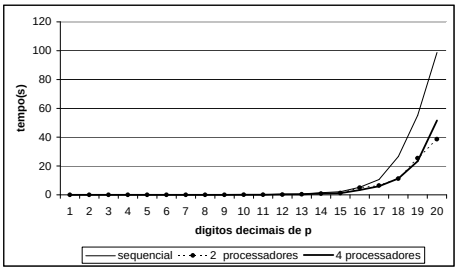

Figura 4: Comparação com os tempos obtidos com 1, 2 e 4 processadores com até 20 dígitos de p

A Tabela 2 mostra os resultados obtidos nos testes realizados com a implementação paralela utilizando 32 processadores, em comparação com o tempo de execução do algoritmo seqüencial. A primeira coluna apresenta o número de dígitos decimais do fator p de n. A segunda coluna mostra o tempo médio gasto no processo de fatoração utilizando o método seqüencial, e a terceira o tempo médio gasto com o método paralelo utilizando 32 processadores.

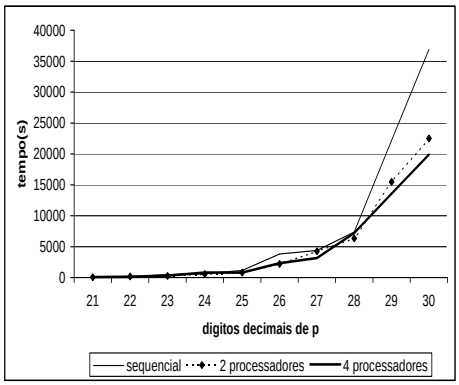

Figura 5: Comparação com os tempos obtidos com 1, 2 e 4 processadores com até 30 dígitos de p

dígitos	Seq.	32 proc.
1	0,000026	0,219407
2	0,000034	0,206337
3	0,000073	0,206567
4	0,000367	0,230233
5	0,003787	0,0207253
6	0,005339	0,0230950
7	0,008670	0,222275
8	0,020742	0,0209557
9	0,052164	0,0213478
10	0,070426	0,213776
11	0,111741	0,231777
12	0,455442	0,236963
13	0,616726	0,259241
14	1,475478	0,295304
15	2,184925	0,331403
16	5,290828	0,561639
17	10,597437	0,776319
18	26,628722	1,218187
19	55,132109	2,562103
20	98,740049	4,195817
21	144,818884	11,395933
22	216,960636	12,095567
23	474,088544	32,333779
24	614,346255	99,693945
25	1155,149328	86,222537
26	3806,466526	436,659065
27	4383,764706	466,594712
28	7352,177194	1148,209806
29	22194,200180	2442,201017
30	63906,148132	5096,811717

Tabela 2: Tempo de Execução

Os dados mostrados na Tabela 2 também podem ser vizualidados através dos gráficos das figuras 6 e 7. Nessas figuras, os resultados estão divididos em testes para fatores de até 20 dígitos e em testes para fatores de 21 a 30 dígitos decimais.

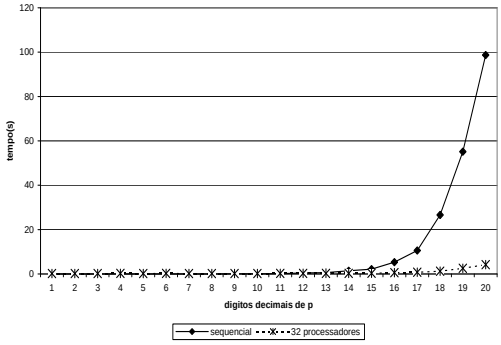

Figura 6: Tempo de execução utilizando 1 e 32 processadores para fatores de 1 a 20 dígitos decimais

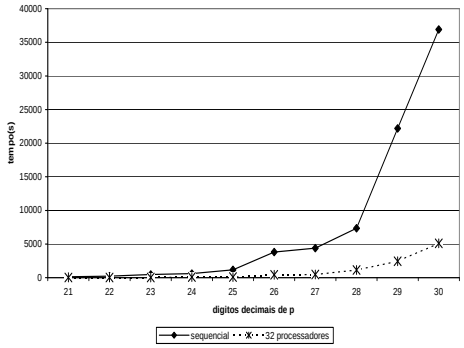

Figura 7: Tempo de execução utilizando 1 e 32 processadores para fatores de 21 a 30 dígitos decimais

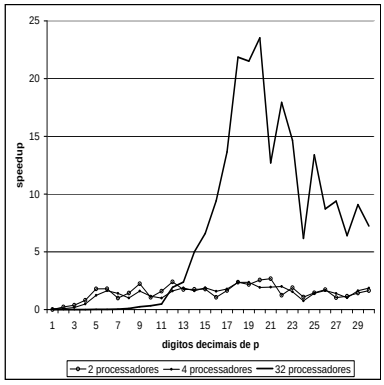

Figura 8: Comparação do *speedup* obtido utilizando 2, 4 e 32 processadores

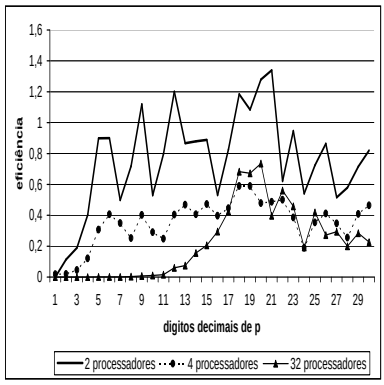

Figura 9: Eficiência obtida com 2, 4 e 32 processadores

Podemos verificar o ganho de desempenho obtido com a implementação paralela quando comparada com a implementação sequencial. Embora a implementação paralela tenha apresentado resultados ineficientes para números compostos de fatores de até 11 dígitos decimais, a partir desse limite obteve um aumento significativo na eficiência.

Resultados do *speedup* e da eficiência obtidos com a implementação paralela para 2, 4 e 32 processadores, podem ser visulalizados nas figuras 8 e 9. Nessas figuras, percebe-se em alguns resultados uma eficiência maior do que 1, embora, pela teoria, esse valor devesse ser no máximo 1. Acredita-se que esse resultado ocorreu devido à natureza probabilística do método ECM. Como o intervalo de números utilizado para obtenção do ponto e da curva foi dividido para cada processador, um processador pode gerar uma curva satisfatória para a fatoração em tempo muito menor do que os demais processadores.

Também é notável nessas figuras que a implementação utilizando 32 processadores apresentou resultados insatisfatórios para fatores de até 11 dígitos decimais. Isso deve-se ao fato de que o tempo gasto com a distribuição dos valores n e k entre os processadores e com a comunicação e sincronização no passo 8, é maior do que o tempo gasto com processamento. No entanto, ficou evidente o aumento na eficiência para fatores acima de 11 dígitos, atingindo o pico de eficiência na fatoração de números com fatores de 20 dígitos decimais.

Calculando a média geral do tempo de execução para cada conjunto de testes (2, 4 e 32 processadores), pode-se verificar o ganho de desempenho obtido com a implementação paralela quando comparada com a implementação sequencial. Para 2 e 4 processadores obteve-se uma melhoria no tempo de execução de $31,9\%$ e $37,6\%$, respectivamente. Ao utilizar 32 processadores o ganho em desempenho foi ainda mais expressivo, representando uma economia de $87,3\%$ no tempo de execução, confirmando assim a melhoria no funcionamento do método ECM através da utilização da computação paralela.

5 Considerações finais

O problema de fatoração de inteiros figura entre os mais importantes problemas matemáticos que ainda não foram solucionados, pois até o momento não se conhece nenhum método que fatore qualquer inteiro dado. Dentre os métodos de fatoração existentes, o método ECM apresenta-se como um dos mais rápidos da atualidade.

Esse trabalho realizou o estudo do método

ECM em relação à duas implementações do mesmo sendo uma sequencial e outra paralela. Foram testados números compostos por 2 fatores de 1 a 30 dígitos decimais.

Os testes realizados mostraram um ganho de desempenho expressivo obtido com a implementação paralela atingindo uma economia média de $87,3\%$ no tempo de execução.

Como trabalhos futuros, sugere-se realizar testes para fatores com um maior número de dígitos e o estudo de algoritmos eficientes para a operação de adição de pontos na curva, que é a principal operação realizada pelo método ECM.

Referências

[1] E. N. Cáceres, H. Mongelli e S. W. Song, Algoritmos Paralelos Usando CGM/PVM/MPI: Uma Introdução, em "XXI Congresso da Sociedade Brasileira da Computação", Jornada de Atualização em Informática, 219-278, 2001.

[2] C. Cardoso, "Fatoração de Números Inteiros Usando Curvas Elíticas", Dissertação de Mestrado, Departamento de Computação e Estatística, UFMS, 2003.

[3] S. Coutinho, "Números Inteiros e Criptografia RSA", IMPA-SBM, 2000.

[4] R. Crandall e C. Pomerance,"Prime Numbers - A Computational Perspective", Springer-Verlag, 2002.

[5] M. O. Dias e A. B. P. Molgora, Estudo do Método Ecm em Coordenadas Afins e Projetivas, em "XXX CNMAC - Congresso Nacional de Matemática Aplicada e Computacional", 2007.

[6] A. Goncalves, "Introducão à Álgebra", IMPA, 1979.

[7] T. Granlund, "GMP: The GNU Multiple Precision Arithmetic Library", Disponível *on-line* em março de 2006 na URL http://www.swox.com/gmp, 2005.

[8] J. Jaja, "An introduction to parallel algorithms", Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1992.

[9] H. Lenstra, Factoring Integers with Elliptic Curves, em "Annals of Mathematics", 126,649-673, 1987.

[10] H. K. Lenstra e J. H. W. Lenstra e M. S. Manasse e J. M. Pollard, The Number Field Sieve, em "Proc. 22nd Annual ACM Symp. On Theory of Computing", 564-572, 1990.

[11] A. B. P. Molgora e E. S. Freitas e F. S. de Paula, Sugestão de um parâmetro ótimo para fatoração de números inteiros, em "XXVI Congresso da Sociedade Brasileira de Computação - SEMISH", 391-402, 2006.

[12] , P. L. Montgomery, Speeding the Pollard and elliptic curve methods of factorization, *Mathematics of Computation*, 48, 243-264, 1987.

[13] M. Snir e S. Otto e S. Huss-Lederman e D. Walker e J. Dongarra, "MPI: The Complete Reference", The MIT Press, 1996.

[14] J. Pollard, Theorems on Factorization and Primality Testing, em "Proceedings of the Cambridge Philosophical Society", 76, 521-528, 1974.

[15] C. Pomerance, The Quadratic Sieve Factoring Algorithm, em "Eurocrypt'84", 209, 169-182, 1985.

[16] R. L. Rivest e A. Shamir e L. Adleman, "On digital signatures and public key cryptosystems", Relatório técnico, MIT/LCS/TR-212, MIT Lab. for Computer Science, 1977.

[17] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", John Wiley & Sons, 1996.

[18] J.H. Silverman, "The Arithmetic of Elliptic Curves", Springer-Verlag, 1986.

[19] J.H. Silverman e J. Tate, "Rational Points on Elliptic Curves", Springer-Verlag, 1992.

[20] H. Suyama, "Informal preliminary report (8)", personal communication to Brent, 1985.