Utilização de Times Assíncronos na Solução do Problema de Cobertura de Conjuntos

Edison L. Bonotto[1], Marcelo Lisboa Rocha[2.]

[1] Programa de Pós-graduação em Informática
Universidade Federal da Paraíba (UFPB) 58059-900 – João Pessoa – PB - Brazil

[2] Programa de Pós-graduação em Modelagem Computacional de Sistemas
Universidade Federal do Tocantins (UFT) 77001-090 – Palmas – TO - Brazil

edisonbonotto@gmail.com, marcelolisboarocha@gmail.com

Resumo. *Este artigo descreve a utilização de métodos heurísticos combinados executados em paralelo (times assíncronos) como forma de resolver o Problema da Cobertura de Conjuntos (PCC). Dado a obtenção da solução ótima do PCC é NP-Difícil, o uso de métodos heurísticos combinados torna possível a resolução do PCC pela obtenção de soluções aproximadas com tempo de processamento aceitável.*

1. Introdução

O Problema da Cobertura de Conjuntos (PCC) consiste em encontrar soluções que minimizem o problema de cobrir um conjunto de necessidades com o menor custo possível, ou seja, encontrar o mínimo de subconjuntos que contenham todos os elementos de um conjunto dado, com fim de minimizar algum valor e, é utilizado numa grande gama de aplicações da vida real, tais como: escalonamento de tripulação, projetos de redes entre outros (LOPES, 1995). Entretanto, a aplicação dos métodos exatos existentes tornam-se inviáveis para instâncias de grandes dimensões, devido ao tempo de processamento necessário para encontrar a solução, o qual cresce exponencialmente.

O trabalho proposto visa encontrar soluções para o PCC utilizando diversas heurísticas (Times Assincronos) executando paralelamente em diversas unidades de processamento (cluster computacional) através de uma biblioteca de passagem de mensagens (MPI). Esta configuração visa à redução do tempo de processamento e a melhoria dos resultados obtidos em relação às heurísticas utilizadas individualmente.

2. O Problema da Cobertura de Conjuntos

O Problema da Cobertura de Conjuntos (PCC) é um problema de análise combinatória e é considerado NP-difícil, isto é, segundo LOPES (1995), não existem algoritmos que os resolvam de forma exata em tempo polinomial.

O PCC tem como entrada um universo “U” de “n” elementos e uma coleção “S” de subconjuntos de “U” cada um com um custo. A saída é uma coleção dos subconjuntos de “S” de tal forma que sua união resulta “U”. A solução, portanto, consiste em encontrar os subconjuntos que contenham todos os elementos de “U”, com fim de minimizar o custo total.

Uma aplicação prática para o problema seria, no caso de uma empresa de transporte coletivo, descobrir quais linhas regulares de ônibus cobrem todas as paradas,

representadas na Figura 1, com o menor custo possível. Transformando estas informações em uma matriz de incidência A_{ij}, onde as colunas *j* da matriz representam as linhas de ônibus e as linhas *i*, as paradas cobertas por elas e acrescentando um custo para cada linha, tem-se a respectiva matriz de incidência demonstra na Tabela 2.

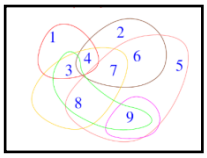

Figura 1. Conjuntos representando linhas de ônibus

Tabela 1. Custo das colunas

	Conjuntos					
	1	2	3	4	5	6
Custo	2	5	6	8	3	4
Identificação	Vermelho	Amarelo	Verde	Marrom	Lilás	Rosa

Tabela 2. Matriz de incidência

Paradas	1	2	3	4	5	6
1	1	0	0	0	0	0
2	0	0	0	1	0	0
3	1	1	1	0	0	0
4	1	1	0	1	0	0
5	0	0	0	0	1	0
6	0	0	0	1	1	0
7	0	1	0	1	1	0
8	0	1	1	0	1	0
9	0	0	1	0	1	1

→Linhas

A solução do Problema de Cobertura de Conjuntos busca por um subconjunto de colunas (conjuntos) com custo mínimo, de forma que cada linha seja coberta, ao menos, por uma coluna (conjunto).

3. Resolução do Problema

A técnica utilizada para a resolução do problema em questão foi um time assíncrono, sendo composto por diversas heurísticas trabalhando em conjunto como um time, visando uma melhora nos resultados em relação às heurísticas individuais e um tempo de processamento menor que a soma dos tempos das heurísticas sendo executadas sequencialmente em separado (PEIXOTO, 1995). No time assíncrono proposto foram empregadas 03 heurísticas - as quais serão descritas no item 3.2: Uma heurística construtiva – baseada na heurística gulosa probabilística de Chvàtal, uma heurística de busca local – baseada na heurística de busca local proposta por FEO (1995) e uma heurística de consenso. O paralelismo vem da execução de cada componente do time (heurística) em uma máquina diferente e propiciando a comunicação entre as mesmas para troca de informações visando obter boas soluções com baixo tempo computacional.

Para verificar a validade da técnica proposta, foram utilizadas instâncias da OR-Library mantida por Beasley e comparações com resultados obtidos executando os algoritmos em apenas um processador, executando em ambiente multiprocessado e com os resultados previamente publicados na literatura. A Tabela 3 mostra as características das instâncias-teste, onde as instâncias são os arquivos que contêm os dados e a dimensão cita o numero de linhas e colunas existentes. Já a densidade representa a esparcidade dos dados, ou seja, representa o percentual de interseções entre os conjuntos, sendo que os menos densos contém um menor número de interseções.

Tabela 3. Características das instâncias-teste

Classe	Dimensão	Densidade	Instâncias
Scp4	200 x 1000	2%	scp4.1 a scp4.9
Scp5	200 x 2000	2%	scp5.1 a scp5.9
Scp6	200 x 1000	5%	scp6.1 a scp6.5
Scpa	300 x 3000	2%	scpa.1 a scpa.5
Scpb	300 x 3000	5%	scpb.1 a scpb.5
Scpc	400 x 4000	2%	scpc.1 a scpc.5
Scpd	400 x 4000	5%	scpd.1 a scpd.5
Scpe	500 x 5000	10%	scpe.1 a scpe.5

3.1. Algoritmos Paralelizados

Diante dos dados com informações de objetos, conjuntos e seus respectivos pesos, será aplicado um algoritmo na máquina mestre, que distribuirá o processamento entre as heurísticas existentes.

Em um primeiro momento, a busca pela solução do problema será realizada pela Heurística Gulosa Probabilística. A solução encontrada é enviada para heurística de busca local para ser novamente processada. Foi definida uma memória com capacidade de armazenamento de 20 (vinte) posições para armazenar as soluções. A cada 05 (cinco) soluções armazenadas nesta memória, será executada a Heurística de Consenso.

A Figura 2 ilustra os passos descritos anteriormente.

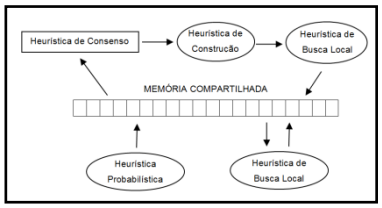

Figura 2. Esquema do método empregado na solução do problema

3.2. Heurísticas Utilizadas

3.2.1. Heurística Gulosa Probabilística

A Heurística Gulosa Probabilística usada foi baseada em uma variação da Heurística Gulosa de Chvàtal, apresentada por FEO (1988). Esta adaptação visa dar maior diversidade às soluções em relação à proposta original.

No caso, cada coluna é analisada levando em conta o número de objetos cobertos e o respectivo custo da coluna. Assim, calcula-se a razão entre o custo da coluna pela cardinalidade (número de objetos cobertos) do conjunto. A coluna é escolhida aleatoriamente entre as que apresentarem a razão igual ou superior ao produto de α e a maior razão entre os conjuntos analisados. α é selecionado aleatoriamente entre 0.6, 0.7, 0.8 e 0.9. Estes valores foram escolhidos, conforme ensaios realizados por FEO (1988).

O algoritmo entra em um laço e a cada coluna acrescentada é verificado se todas as colunas escolhidas até aqui geram uma solução válida. Ao encontrar a solução válida, o laço é interrompido.

Esta solução válida não será obrigatoriamente a melhor solução. Então, para um melhor resultado, o algoritmo é executado 20 vezes e a melhor solução é enviada ao mestre para ser processada. Este número de repetições foi escolhido levando-se em

conta o tempo de processamento e a melhora apresentada nas soluções encontradas, devendo ser revisto dependendo da distância necessária entre as soluções encontradas e as soluções ótimas para cada problema específico.

3.2.2. Heurística de Busca Local

O algoritmo de busca local é aplicado na solução encontrada pela heurística gulosa probabilística, com a intenção de melhorar o resultado obtido. Nesta implementação, cada k-tupla encontrada na solução é comparada com os conjuntos existentes verificando qual conjunto cobre todos os objetos cobertos pela k-tupla e calcula o custo. Caso o custo do novo conjunto seja inferior ao custo da k-tupla e este ainda não faça parte da solução e cubra, no mínimo, os objetos cobertos pela k-tupla, esta é substituída por este conjunto, melhorando desta maneira o custo total.

A fim de diminuir o tempo de processamento, nem todos os conjuntos são comparados e a seguinte estratégia na verificação dos conjuntos foi adotada: A quantidade de conjuntos utilizados para a comparação depende do número de conjuntos encontrados no problema. As k-tuplas são comparadas com os conjuntos presentes num vetor sequencialmente, mas dependendo do número de conjuntos, alguns destes não são verificados, avançando no vetor em saltos maiores que 1. A fim de termos uma melhor diversidade dos conjuntos verificados, a cada iteração é escolhido aleatoriamente de quantos em quantos saltos os conjuntos são verificados na comparação sequencial. Para instâncias com até 1000 conjuntos, todos são comparados; de 1000 até 1999, o intervalo varia de 1 a 5; de 2000 até 3999, o intervalo varia de 1 a 3; e, instancias com mais de 4000 conjuntos, o intervalo varia de 1 a 7. Para definir o intervalo máximo dos saltos foram realizados testes nos dados constantes na OR-library – Tabela 03 (Características das instâncias-teste). Usando estratégia análoga, nem todas as k-tuplas são comparadas. Não foi possível estabelecer um valor ideal para todos os problemas, pois o tempo depende, também, das características individuais dos dados de entrada.

3.2.3. Heurística de Consenso

A heurística de consenso tem como finalidade escolher as possíveis melhores opções de conjuntos entre as soluções apresentadas pelas heurísticas durante o processamento. Acredita-se que os conjuntos escolhidos fazem parte da solução ótima. Esta heurística é executada a cada 05 inserções ou alterações na memória de soluções.

A heurística funciona da seguinte maneira: A memória que armazena as soluções potenciais é analisada e, os conjuntos de maiores cardinalidade (número de vezes) presentes nas soluções farão parte de uma solução parcial. Neste experimento, todos os conjuntos que apresentarem cardinalidade igual ou superior a 80% (oitenta) das soluções já armazenadas na memória, serão selecionados para fazerem parte de uma solução parcial. Esta solução parcial não necessariamente cobrirá todos os conjuntos.

A solução parcial será, então, completada pela Heurística Gulosa Probabilística - descrita no item 3.2.1. - a fim de que se obtenha uma solução completa, ou seja, com a cobertura de todos os conjuntos.

No resultado obtido pela Heurística Gulosa Probabilística será aplicada à heurística de Busca Local – descrita no item 3.2.2., visando o aprimoramento do resultado. Ao final do processamento, a solução é enviada para o mestre para processamento.

3.2.4. Mestre

O mestre não é exatamente uma heurística e sim, uma parte do código que tem como função efetuar a chamada às heurísticas e processar o retorno destas, além de controlar a memória de soluções. Tem também como função distribuir as diversas heurísticas pelos hosts disponíveis.

Inicialmente, é chamada a heurística probabilística. A solução retornada é comparada com as existentes na memória de soluções. Caso exista solução idêntica a armazenada, a solução é descartada; caso contrário, se a memória não estiver completa, a solução será armazenada em uma nova posição. No caso da memória estar completa, é verificado se o custo da solução é melhor que o custo da pior solução armazenada. Caso positivo, a solução é armazenada no lugar desta. Cada resultado retornado da heurística probabilística é, também, enviado para a heurística de busca local e o resultado desta, segue os mesmos critérios descritos acima para armazenamento na memória.

A cada 5 inserções ou alterações realizadas na memória, uma chamada à heurística de consenso é efetuada. O resultado retornado desta, também segue os mesmos critérios já descritos, para armazenamento na memória de soluções.

Nesta implementação paralelizada, o mestre é executado em um computador dedicado e cada uma das heurísticas – probabilística, busca local e consenso - são distribuídas em computadores distintos, totalizando 04 hosts.

4. Testes e Resultados

Nesta seção, serão apresentados os resultados computacionais, realizando um comparativo na qualidade das soluções entre os algoritmos sendo executados paralelamente com os algoritmos de Busca Local, a Heurística Probabilística e os resultados encontrados na literatura. Este comparativo tem como finalidade validar a eficácia do uso do paralelismo na solução do problema apresentado.

O time assíncrono proposto neste trabalho foi implementada fazendo uso da Linguagem C de programação e compilado com *gcc 4.8* utilizando a opção -O3 do compilador. Os testes foram executados em 4 máquinas com a seguinte configuração: Pentium Core Duo de 2,5 Ghz, 4 Gb de RAM e sistema operacional Linux Ubuntu 12.4.

Os resultados foram obtidos realizando uma bateria de 3 testes (com exceção dos resultados das instâncias scp41, scp51, scp61, scpa1, scpb1, scpc1, scpd1 e scpe1, as quais foram realizados 05 testes) sobre cada uma das instâncias da OR-Library listadas na tabela 5. As soluções ótimas conhecidas foram extraídas de BEASLEY (1990).

A tabela 5 consolida os resultados obtidos e contem as seguintes informações: Primeira coluna (dados): corresponde ao nome da instância utilizada para o teste; Segunda coluna (melhor solução conhecida): contem os melhores resultados encontrados na literatura – no caso, soluções ótimas – e foram extraídas de BEASLEY (1990); As próximas 04 (quatro) colunas – processamento paralelo – mostram os resultados obtidos na solução proposta neste trabalho. A coluna “melhor solução” mostra a melhor solução – menor custo - dentre todas as solução encontradas no processamento; a coluna “gap %” mostra a distância entre o resultado obtido e a melhor solução encontrada na literatura; já a próxima coluna mostra o tempo médio de processamento entre as n execuções na mesma instância e a coluna “quant. melhor solução” mostra quantas vezes a melhor solução encontrada aparece nos resultados; As colunas “processamento

probabilística" e "processamento busca local", mostram, nesta ordem, a melhor solução encontrada, o tempo médio de processamento e quantas vezes a melhor solução aparece nos resultados quando a heurística gulosa probabilística e a heurística de busca local são executadas isoladamente, sem troca de dados entre si. Estes resultados servem para comparar com os resultados da solução proposta e validar sua eficácia.

Analisando os resultados da Tabela 5, pode-se notar que a distância entre os resultados obtidos no método proposto e as melhores soluções encontradas na literatura são muito próximos, variando de 0,00% a 6,57%, com uma média de 2,03% e uma mediana de 2%, onde a distância 0,00% significa que encontrou uma solução ótima (solução com o menor custo possível). Foram encontradas 07 (sete) soluções ótimas. Tudo isto nos faz crer que as soluções são viáveis para muitas aplicações práticas.

Já na comparação com os resultados das heurísticas sendo executadas individualmente, em todas as ocasiões a solução proposta encontrou resultados melhores, com exceção apenas da classe scpe, a qual contém instâncias com baixa complexidade de processamento, onde os resultados foram iguais – nunca piores. O tempo de processamento do time assíncrono foi aproximadamente 50% inferior a soma do tempo gasto pelas 02 (duas) heurísticas sendo executadas sequencialmente.

Tabela 5. Resultados obtidos

Dados	Melhor Solução Conhecida	Processamento Paralelo				Processamento Probabilística			Processamento Busca Local		
		Melhor solução	Gap %	Tempo Médio (Min)	Quant melhor Solução	Melhor solução	Tempo Médio (Min)	Quant melhor Solução	Melhor solução	Tempo Médio (Min)	Quant melhor Solução
scp41	429	434	1,17	22,48	1	452	2,18	9	440	57,10	1
scp42	512	527	2,93	22,31	1	569	2,17	1	552	53,63	1
scp43	516	532	3,10	21,06	2	576	2,13	1	549	40,54	1
scp44	494	504	2,02	22,11	1	537	2,17	2	518	40,71	1
scp45	512	519	1,37	20,75	1	553	2,13	1	524	31,02	1
scp46	560	568	1,43	19,71	3	591	2,09	1	598	38,87	1
scp47	430	434	0,93	15,71	1	464	1,96	1	450	30,67	1
scp48	492	493	0,20	15,48	1	514	1,96	2	525	31,06	1
scp49	641	667	4,06	23,86	1	715	2,22	1	720	56,02	1
scp51	253	260	2,77	49,64	2	278	7,63	2	272	131,88	1
scp52	302	315	4,30	77,61	2	334	7,61	6	321	102,96	1
scp53	226	230	1,77	42,24	9	241	7,11	1	232	76,51	1
scp54	242	247	2,07	48,64	1	255	7,44	1	256	128,12	1
scp55	211	213	0,95	44,71	1	226	7,33	1	224	105,34	1
scp56	213	227	6,57	43,63	3	233	7,28	1	290	117,70	1
scp57	293	296	1,02	45,94	5	308	7,45	3	432	120,29	1
scp58	288	291	1,04	48,47	1	309	7,49	11	414	118,12	1
scp59	279	280	0,36	47,43	2	295	7,10	1	386	115,85	1
scp61	138	143	3,62	4,72	1	151	1,27	1	150	9,84	1
scp62	146	149	2,05	4,87	1	160	1,27	2	164	6,81	1
scp63	145	148	2,07	3,78	2	151	1,21	2	161	7,43	1
scp64	131	135	3,05	4,49	3	136	1,28	1	137	9,04	1
scp65	161	169	4,97	5,33	3	187	1,32	3	183	8,37	2
scpa1	253	258	1,98	171,03	1	271	20,43	2	265	396,47	1
scpa2	252	257	1,98	230,00	1	271	20,60	2	265	517,41	1
scpa3	232	234	0,86	177,28	1	249	20,17	1	255	475,73	1
scpa4	234	241	2,99	190,50	3	256	20,51	1	255	481,93	1
scpa5	236	239	1,27	194,72	1	255	20,66	1	248	685,21	1
scpb1	69	71	2,90	26,60	11	71	10,69	6	87	59,27	1
scpb2	76	76	0,00	36,14	1	80	11,76	1	87	71,51	1
scpb3	80	81	1,25	32,87	3	82	11,21	6	84	64,01	1
scpb4	79	81	2,53	40,76	1	87	11,95	6	90	81,33	1
scpb5	72	74	2,78	31,46	8	76	11,37	2	81	69,43	1
scpc1	227	234	3,08	413,36	1	248	43,52	1	244	947,68	1
scpc2	219	224	2,28	419,71	1	239	43,86	1	238	913,08	1
scpc3	243	244	0,41	376,76	1	263	40,42	2	260	999,07	1
scpc4	219	232	5,94	406,44	2	241	42,22	1	240	1022,44	1
scpc5	215	220	2,33	328,34	2	229	41,74	4	229	927,95	1
scpd1	60	62	3,33	54,07	6	66	22,13	2	71	119,31	1
scpd2	66	68	3,03	47,74	7	69	22,09	11	76	112,59	1
scpd3	72	73	1,39	58,77	3	76	23,21	1	82	114,97	2
scpd4	62	62	0,00	58,50	1	64	23,20	5	67	112,58	1
scpd5	61	63	3,28	57,09	4	64	23,10	1	77	110,47	1
scpe1	5	5	0,00	0,03	17	5	0,04	13	6	0,02	4
scpe2	5	5	0,00	0,03	9	5	0,04	10	5	0,02	1
scpe3	5	5	0,00	0,03	16	5	0,04	10	6	0,02	3

scpe4	5	5	0,00	0,04	10	5	0,05	9	6	0,03	7
scpe5	5	5	0,00	0,03	14	5	0,05	11	6	0,02	4

5. Conclusões

Observando os resultados obtidos, pode-se verificar que o uso da técnica de times assíncronos melhora sensivelmente a qualidade das soluções encontradas e reduz o tempo de processamento em relação à soma das heurísticas executadas sequencialmente. A solução paralelizada encontrou soluções melhores ou iguais em todos os casos (instâncias de testes utilizadas), não havendo qualquer resultado com pior solução em nenhum dos testes realizados. As soluções de menor custo são sempre melhores ou, em poucas ocasiões, iguais que as propiciadas pela solução monoprocessada (sequencial).

Na comparação com os melhores resultados obtidos na literatura, as soluções são promissoras, sendo aceitáveis na maioria dos problemas. Todos os resultados ficaram a uma distância próxima daquelas e, em alguns casos, uma solução ótima foi encontrada.

A utilização de agentes baseados nos algoritmos de Chvàtal e busca local, combinados com um algoritmo de consenso, sendo executados em ambiente paralelo, retornaram resultados melhores do que os obtidos com esses mesmos algoritmos sendo executados individualmente em ambiente monoprocessado. Contudo, maiores estudos e novos desenvolvimentos podem ser feitos de modo a buscar uma eficiência ainda maior para o time assíncrono proposto. Como principais sugestões, têm-se:

- Pode-se substituir ou aprimorar os algoritmos de construção, busca local e consenso, objetivando melhores resultados.
- Determinar o numero de iterações que o algoritmo da heurística gulosa probabilística deve ser executado, a fim de se obter um melhor custo benefício.
- Realizar testes a fim de determinar, no algoritmo de consenso do time assíncrono, o melhor percentual que indica o número mínimo de vezes que um valor deve aparecer repetido em todas as soluções para ser considerado comum e aparecer na solução final.

Referências

BEASLEY, J. E. e JORNSTEN, k. Enhancing an Algorithm for Set Covering Problems. European Journal of Operational Research, London, England, 1990.

FEO, Thomas A. et al. A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. The University of Texas, Austin – TX, USA, revised October 1988.

FEO, Thomas A. e RESENDE, M. G. C. Greedy Randomized Adaptive Search Procedures. Kluwer Academic Publishers, Boston, USA, 1995.

LOPES, Luciana de Sousa. Uma Heurística Baseada em Algoritmos Genéticos Aplicada ao Problema de Cobertura dos Conjuntos. Dissertação de M.Sc, INPE, 1995.

PEIXOTO, H. P. Metodologia de Especificação do Time Assíncrono Para Problemas de Otimização Combinatória. Dissertação de M.Sc., UNICAMP, Campinas, SP, Brasil, 1995.

ROCHA, M. Lisboa. Aplicações de Algoritmos Paralelos e Hibridos para o Problema de Árvore de Steiner Euclidiana no Rn. Tese de doutorado, Rio de Janeiro, 2008.