POPMUSIC aplicada ao Problema de Posicionamento de Réplicas e de Distribuição de Requisições em Redes de Distribuição de Conteúdos.

Ronaldo Goldbach, Adriana C. F Alvim
Universidade Federal do Estado do Rio de Janeiro
Av. Pasteur 458, Urca, Rio de Janeiro – RJ
ronaldo.goldbach@uniriotec.br, adriana@uniriotec.br

Tiago A. Neves
Universidade Federal Fluminense
Av. dos Trabalhadores 420, Volta Redonda - RJ
tneves@id.uff.br

RESUMO

Redes de computadores conectados através da internet (RDCs) têm sido usadas para atender à crescente demanda de conteúdos digitais. Para manter os custos no nível mais baixo possível, é preciso decidir em que servidores posicionar réplicas de quais conteúdos, e a que servidor direcionar cada requisição. Usualmente, não se sabe *a priori* quando nem quais conteúdos serão requisitados; e não é viável física nem economicamente replicar todos os conteúdos em todos os servidores. Os diversos arranjos possíveis caracterizam problemas combinatórios para os quais é difícil obter soluções de boa qualidade em tempo computacional razoável. Este trabalho apresenta o algoritmo POP-PPRDR, que aplica o método de decomposição POPMUSIC (*Partial OPtimization Metaheuristic Under Special Intensification Conditions*) ao Problema de Posicionamento de Réplicas e de Distribuição de Requisições em RDCs. Experimentos computacionais mostraram que POP-PPRDR obteve soluções de melhor qualidade quando comparado com outro método recentemente proposto para resolver o PPRDR.
PALAVRAS CHAVE. Meta-heurística, Métodos de decomposição, POPMUSIC, PPRDR, Redes de Distribuição de Conteúdos.

Área principal Meta-heurísticas.

ABSTRACT

Internet-connected computer networks (CDNs) have been used to cope with the growing demand for digital contents. To keep costs as low as possible, it is important to decide what content will be kept on what server, and what request will be directed to what server. Usually, one does not know *a priori* what will be the demand, nor when will it happen; and it is not physically nor economically feasible to replicate every content in every server. The different possible arrangements characterize a combinatorial problem to which a good-quality computer-based solution is very hard to reach in a reasonable time. This article presents the algorithm POP-PPRDR, which applies POPMUSIC (Partial OPtimization Metaheuristic Under Special Intensification Conditions) to the Replica Placement and Requisition Distribution Problem (RPRDP) in CDNs. Computational experiments showed that POP-PPRDR was able to obtain better quality solutions when compared to a recently tested method used to solve RPRDP.
KEYWORD. Metaheuristics, Content Distribution Network, Decomposition methods, POPMUSIC, RPRDP.

Main area Metaheuristics.

1 Introdução

Quando uma pessoa usa a internet para ver um videoclipe no seu computador ou baixar uma versão de um programa antivírus, ela está gerando uma demanda pelo serviço de *download* de um conteúdo. As demandas, muitas vezes, não são previsíveis e podem coincidir, no tempo, com demandas de outras pessoas que pretendem fazer *download* a partir de um mesmo computador ("servidor") onde o conteúdo fica armazenado. Em alguns casos, o servidor pode ficar sobrecarregado: sua banda (volume máximo de dados transmissível pela ligação entre o servidor e seus clientes) pode ser insuficiente para atender simultaneamente a todas as requisições feitas. Muitos estudos têm sido realizados para minimizar o tráfego na rede, respeitando restrições como capacidade de armazenamento de servidores, Qualidade de Serviço (QoS – Quality of Service) negociada com alguns clientes – por exemplo na forma de um tempo máximo de atendimento, ou de uma banda mínima que tem que ser disponibilizada, etc. [Neves et al. (2008), Neves (2011)].

De acordo com Bakiras e Loukopoulos (2005), Tenzakhti et al., (2004), e Zhou e Xu, (2007), o problema acima pode ser modelado pelo uso de Redes de Distribuição de Conteúdos (RDC – em inglês, CDN, *Content Distribution Network*), formadas por servidores virtuais hospedados em servidores de uma topologia real, que armazenam e distribuem o conteúdo demandado. Uma RDC posiciona uma ou mais réplicas de conteúdos em determinados servidores para equilibrar a carga entre eles, minimizar o tráfego e os congestionamentos e evitar ou reduzir atrasos na entrega dos conteúdos requisitados. Manter uma réplica de cada conteúdo em todos os servidores garantiria ter qualquer conteúdo o mais perto possível de cada possível cliente, mas isso pode ser muito caro ou impossível, por restrições da capacidade de armazenamento ou banda de cada servidor e da quantidade máxima permitida de cópias de certos conteúdos etc.

O PPRDR, Problema de Posicionamento de Réplicas e Distribuição de Requisições, tratado por Neves et al. (2008) e Neves (2011), conjuga dois típicos problemas de gerenciamento em uma RDC: o Problema de Posicionamento de Réplicas (PPR), que consiste em decidir em que servidor(es) posicionar cada réplica, com o objetivo de minimizar o tráfego e os custos de transmissão; e o Problema de Distribuição de Requisições (PDR), que consiste em distribuir a demanda entre os servidores, uma vez as réplicas posicionadas, com o objetivo de minimizar os custos de transmissão. No PPRDR procura-se alocar cópias do conteúdo próximo de onde se prevê que sejam requisitados e dirigir as requisições que ocorrem ao(s) servidor(es) mais próximo(s) do cliente (ou com mais banda disponível) que hospede(m) uma réplica do conteúdo solicitado. O PPRDR pertence à classe NP-Difícil [Garey e Johnson (1979), Neves (2011)], cujos problemas têm sido tratados por dois tipos de métodos: exatos e heurísticos. Enquanto os primeiros garantem alcançar a solução ótima para o problema, os métodos heurísticos não oferecem essa garantia; possibilitam, porém, o tratamento de instâncias maiores em tempo menor [Reeves e Beasley (1993)].

Neves (2011) apresenta diversas técnicas de solução para o problema, entre elas uma que emprega métodos heurísticos e meta-heurísticos e que obtém resultados excelentes, quando comparados a outros resultados existentes na literatura, para instâncias com até 100 servidores, mas que não chega a uma conclusão em termos de qualidade da solução para instâncias maiores. Como trabalhos futuros, o autor propõe a utilização de diferentes meta-heurísticas para resolver o PPRDR. No entanto, mesmo considerando a escalabilidade das meta-heurísticas com o tamanho dos problemas, a partir de certo porte a solução pode deixar de ser eficiente: o limite de resolução prático de muitos algoritmos é de instâncias com algumas poucas centenas de itens. Instâncias de problemas combinatórios reais, que podem envolver milhares de itens, são considerados "grandes" [Taillard et al. (2007)] e estão claramente além do limite de centenas de itens [Ostertag et al. (2009)]. Para diminuir a complexidade computacional de abordagens práticas para a resolução de problemas de otimização combinatória grandes, pode-se considerar a decomposição destes problemas em subproblemas. Para realizar tais decomposições, de forma a obter subproblemas compactos e bem estruturados, é aconselhável utilizar métodos relativamente elaborados.

POPMUSIC (*Partial OPtimization Metaheuristic Under Special Intensification Conditions*) [Taillard e Voss (2001)] é um método genérico de otimização especialmente projetado para a otimização de soluções de instâncias grandes de problemas de otimização combinatória.

Este artigo apresenta POP-PPRDR [Goldbach (2013)], uma heurística capaz de prover solução para o PPRDR em RDCs de médio e grande portes, empregando a meta-heurística POPMUSIC.

O restante deste trabalho está assim organizado: a Seção 2 apresenta o método de decomposição POPMUSIC e a Seção 3 uma seleção de trabalhos relacionados ao problema. A Seção 4 detalha a solução proposta para o PPRDR. A Seção 5 relata os experimentos computacionais realizados e a Seção 6 apresenta as conclusões do trabalho e sugestões para próximos passos da pesquisa.

2 O método de decomposição POPMUSIC

POPMUSIC é um método geral projetado para lidar com instâncias grandes de problemas de otimização combinatória difíceis, através de sua decomposição em problemas menores. De acordo com Taillard e Voss (2001), não é fácil descobrir a maneira apropriada de decompor um problema, *a priori*. A decomposição de um problema em subproblemas pode levar a soluções de qualidade apenas moderada, já que os subproblemas podem ter sido criados de maneira arbitrária. POPMUSIC parte de uma solução disponível para o problema (não necessariamente boa), e tenta melhorar a qualidade da solução de partes da solução inicial, *a posteriori*. O pseudocódigo na Figura 1 ilustra um esquema geral para o método POPMUSIC.

Procedimento POPMUSIC (S, r)

1. Dados a solução inicial S, composta pelas partes s_1, ..., s_p ; e r, tamanho dos subproblemas;
2. O = { s_1, ..., s_p }; /* conjunto de controle
3. Enquanto $O \neq \emptyset$ faça
4. Selecione uma parte $s_j \in O$; /* parte semente
5. Monte subproblema R_j composto de r partes de S, incluindo s_j e as r-1 partes mais fortemente relacionadas a s_j ;
6. Submeta R_j ao processo de tentativa de melhoria;
7. Se R_j tiver sido melhorado então
8. Atualize na solução corrente S as r partes do subproblema otimizado R_j;
9. Inclua em O todas as partes do subproblema otimizado R_j que não estejam lá;
10. Senão
11. Remova de O a parte semente s_j;
12. fim Enquanto;

fim POPMUSIC.

Figura 1. Pseudocódigo POPMUSIC baseado em Taillard e Voss (2001).

Sejam S uma solução de um determinado problema que se deseja melhorar e r o tamanho escolhido para os subproblemas. Suponha que S possa ser definida como um conjunto de p partes s_1, ..., s_p (linha 1), e que uma medida de relacionamento possa ser definida entre quaisquer duas partes. Na linha 4, POPMUSIC seleciona uma parte s_j, à qual chama parte semente. Na linha 5, são selecionadas r-1 < p partes mais fortemente relacionadas a s_j para formar um subproblema R_j de tamanho r. Para evitar gerar o mesmo subproblema mais do que uma vez, implementa-se um procedimento de controle: um conjunto O armazena as partes que podem ser usadas como sementes para definir um subproblema que possa ser tratado em busca de melhora. O conjunto O é inicializado na linha 2. O subproblema é submetido a uma tentativa de melhoria (linha 6), por algum método, exato ou heurístico. Quando O estiver vazio, então todos os subproblemas terão sido examinados, não restando algum que possa ser melhorado, e o processo

acaba (linhas 3 e 12). Durante a aplicação do algoritmo, o subproblema R_j pode ter sido melhorado (linha 7). Nesse caso, algumas de suas partes terão sido alteradas e a solução S é atualizada (linha 8). Com a alteração das partes, novas melhorias podem ser possíveis em subproblemas criados a partir daquelas partes; logo, tais subproblemas devem ser reexaminados e, portanto, as partes que eventualmente tenham sido retiradas anteriormente são reinseridas no conjunto O (linha 9). Caso não tenha sido possível melhorar o subproblema, a parte semente correspondente é retirada de O (linha 11). Se a definição das partes e subproblemas são feitas de maneira adequada, a cada melhora obtida para o subproblema corresponde uma melhora no problema completo.

Existem algumas variantes possíveis para o método POPMUSIC. Uma variante mais rápida, ao invés de remover de O a parte semente s_j (linha 11), remove todas as partes de R_i. Por outro lado, uma variante mais lenta, na linha 9, incluiria todas as partes do problema original e não apenas as partes de R_i.

Alvim e Taillard (2013) lembram que a complexidade do método pode ser muito alta, uma vez que O não é obrigatoriamente reduzido a cada iteração. Entretanto, diversas implementações mostram de forma empírica que o número de iterações de um algoritmo baseado neste procedimento cresce quase linearmente com p. Nos experimentos relatados na Seção 5, verificou-se que a quantidade média de iterações foi sempre em torno de duas vezes o tamanho da instância (a média das quantidades de iteração / tamanho da instância foi de 2,18).

Além do tamanho dos subproblemas (o parâmetro r), a meta-heurística POPMUSIC prevê que os seguintes componentes sejam especificados de acordo com o problema a resolver: 1) o procedimento para obter uma solução inicial; 2) a definição do que é uma "parte" da solução; 3) a função de relacionamento entre as partes para criar um subproblema; 4) o procedimento de seleção de uma parte como semente que será usada para dar origem a um subproblema; e 5) o procedimento para otimizar os subproblemas.

Este método tem se mostrado altamente eficiente em resolver problemas grandes, conforme demonstrado, por exemplo, nos trabalhos de Alvim e Taillard (2009), Alvim e Taillard (2013) e Ostertag et al. (2009).

3 Trabalhos Relacionados

Na literatura é possível encontrar diferentes abordagens para resolver problemas em RDCs. Das soluções propostas para o PPR, a maioria trata o caso estático, cujo objetivo é achar o melhor posicionamento das réplicas a partir de um padrão de tráfego de requisições conhecido. Embora não reflita a dinâmica da maior parte dos casos reais, tal proposta pode ser usada em casos onde as requisições apresentam pequena ou nenhuma variação, ou para estudar o comportamento de algoritmos. Almeida et al. (2004) tratam o PPR estático e propõem um algoritmo usando uma formulação matemática e várias heurísticas para o problema. O trabalho aponta a necessidade de testes em redes maiores e mais diversas. Não se considera a qualidade percebida pelo cliente (QoS), nem a otimização do posicionamento e roteamento para entrega de múltiplos conteúdos.

Bartolini et al. (2003) abordam o PPR dinâmico, considerando a variação na demanda dos usuários e o desempenho da rede, modelando o problema como um processo de decisão de Markov.

Bektas et al. (2007) abordam os problemas de Dimensionamento e Posicionamento de Servidores (PLS) e o PPRDR' estáticos. O PPRDR' se assemelha ao PPRDR, mas não trata a garantia da QoS. Entre outras soluções, os autores propõem um algoritmo heurístico guloso.

Huang e Abdelzaher (2004) apresentam um mecanismo para estabelecer um limite no tempo de espera entre a solicitação e a disponibilização dos dados, utilizando uma modelagem por grafos e uma abordagem distribuída. O tamanho dos arquivos afeta os resultados, uma vez que cada tamanho tem uma latência e os arquivos solicitados podem ter qualquer tamanho.

Neves (2011) trata o PPRDR considerando, entre outros, o atendimento à garantia da Qualidade de Serviço (QoS); múltiplos conteúdos, de vários tamanhos, inclusive os extensos; e a possibilidade de uma requisição ser atendida em mais do que um período de tempo. Um dos

algoritmos propostos pelo autor combina o método ILSAM (*Iterated Local Search with Adaptive Memory*, que consiste da meta-heurística *Iterated Local Search* [Lourenço et al. (2002)] usando memória adaptativa com a heurística ARTR (*Adaptive Record-To-Record*, que é a RTR com memória adaptativa) e gera soluções de boa qualidade, com redução de custo de até cinco ordens de grandeza comparadas à heurística usada em RDCs reais. No entanto, os resultados obtidos para o PPR em instâncias com mais do que 100 servidores não são conclusivos, devido ao reduzido número de iterações feitas pelos algoritmos.

Shahabi e Banaei-Kashani (2002) pesquisam questões relacionadas à redução de custos de comunicação e armazenamento, desenvolvendo um mecanismo de entrega que envolve o posicionamento das réplicas ao longo dos servidores através de uma hierarquia de servidores com redundância.

Wauters et al. (2006) propõem um conjunto de algoritmos para posicionamento de réplicas baseados em uma formulação de programação linear inteira do problema de posicionamento centralizado. O trabalho apresenta heurísticas que adaptam dinamicamente o posicionamento das réplicas às variações de comportamento dos usuários, disponibilidade de conteúdo e carga na rede.

Zhou e Xu (2007) investigam a replicação de conteúdo, enunciando o problema como um problema de otimização combinatória com os objetivos de maximizar o número de réplicas de cada vídeo e balancear a carga dos servidores, considerando as limitações de capacidade e de largura de banda dos servidores. Os autores propõem uma heurística baseada em *Simulated Annealing* para uma generalização do problema onde um mesmo conteúdo pode ter vários graus de qualidade. Os algoritmos são testados em um cenário com servidores providos de grande capacidade de armazenamento, com grande largura de banda e conteúdos de vídeo extensos (90 minutos).

4 POPMUSIC aplicada ao PPRDR

O presente trabalho propõe o Algoritmo POP-PPRDR para resolver o PPRDR, empregando a meta-heurística POPMUSIC proposta por Taillard e Voss (2001).

A seguir, as escolhas dos componentes livres de POPMUSIC utilizadas no algoritmo POP-PPRDR:

- Solução inicial: Foi selecionada a heurística HNH (*Hybrid Network Heuristic*) proposta por Neves (2011) para gerar a solução inicial de POPMUSIC. HNH apresenta bons resultados na resolução da versão *on-line* do problema, em termos de tempos computacionais e da qualidade das soluções obtidas. HNH é composta pelas melhores soluções encontradas por Neves (2011) para cada um dos PPR e PDR, subproblemas do PPRDR. Para o PPR é usada a AGAP (*Adapted Generalized Allocation Problem*), um método exato que permite a alocação de um conteúdo em mais de um servidor e garante que todo conteúdo tenha ao menos uma réplica. O PDR é modelado como um problema de fluxo em rede, representado como um grafo acíclico e direcionado, e resolvido como um Problema de Fluxo de Custo Mínimo (PFCM). Uma solução é a associação de cada um dos servidores utilizados – pode haver servidores da instância sem utilização – com suas respectivas réplicas de conteúdo e uma parcela (0 a 100%) de atendimento a requisições. Uma requisição pode ser atendida por mais de um servidor.
- Parte da solução: Uma parte s_j foi definida por um servidor j com seus respectivos conteúdos C_{jt} nos períodos de tempo t, para $t = 1, 2, ..., |T|$, e os percentuais de cada uma das requisições que serão atendidas por j (Q_{jt}) nos períodos t, para $t = 1, 2, ..., |T|$. Há uma relação biunívoca entre servidor / parte. A definição de uma parte s_j atribui o atendimento de cada parcela de requisição a um único servidor; as partes são disjuntas, sem interferência entre elas. Por extensão, não ocorrem interferências entre subproblemas. Isso permite dividir a solução inicial, que forma o problema original, em partes que geram subproblemas independentes entre si.
- Relação entre duas partes: A relação entre duas partes foi definida como o inverso da distância entre seus respectivos servidores, isto é, quanto menor a distância entre dois

servidores, mais fortemente estão relacionadas suas respectivas partes. A distância entre cada par de servidores é um dado do problema, expressa pelo RTT (*Roundtrip Time*, tempo para um dado trafegar de um servidor a outro e voltar) médio entre tal par de servidores ao longo de todo o horizonte de planejamento.

- Seleção de semente: As partes disponíveis para servir de semente ficam organizadas num conjunto O (que inicialmente contém todas as partes da solução inicial), tratado como uma fila seguindo a estratégia FIFO (*First In, First Out*). O subproblema R_j é criado com a parte semente s_j e as r-1 partes mais fortemente relacionadas (mais próximas) de s_j.
- Procedimento de melhoria para otimizar subproblemas: Cada subproblema é submetido ao algoritmo de melhoria proposto por Neves (2011) para solução do modelo *off-line* do PPRDR, empregando ILSAM com ARTR, conforme citado na Seção 3. Mais detalhes podem ser encontrados no trabalho de Neves (2011). Diferentemente do que ocorre naquele trabalho, onde o critério de parada é o tempo, neste trabalho o critério de parada utilizado para o ILSAM é baseado na ocorrência de certa quantidade de iterações sem melhora. Mais detalhes podem ser encontrados na Seção 5.2 e no trabalho de Goldbach (2013).

Para a solução do PPRDR, considerou-se a mesma função objetivo descrita em Neves (2011), que possui os seguintes componentes: conjunto Q de requisições a atender; conjunto V de servidores da RDC; conjunto C de conteúdos; conjunto T de períodos de tempo; custo c_{ijt} de atendimento da requisição i pelo servidor j no período t; fração x_{ijt} do conteúdo solicitado pela requisição i entregue pelo servidor j no período t; penalidade p_{it} por usar *backlog* da requisição i no período t; *backlog* b_{it} da requisição i no período t; custo h_{kjlt} de replicar conteúdo k no servidor j a partir do servidor l no período t e indicador w_{kjlt} de cópia do conteúdo k pelo servidor j a partir do servidor l no período t; vale 1 se o conteúdo é copiado, 0 em caso contrário. A função objetivo é mostrada em (1).

$$Min \sum_{i\in Q}\sum_{j\in V}\sum_{t\in T} c_{ijt}x_{ijt} + \sum_{i\in Q}\sum_{t\in T} p_{it}b_{it} + \sum_{k\in C}\sum_{j\in V}\sum_{l\in V}\sum_{t\in T} h_{kjlt}w_{kjlt} \qquad (1)$$

Para reduzir o atendimento de requisições por servidores muito distantes, privilegiando a escolha de servidores que possam atender as requisições dentro do atraso máximo estipulado, a formulação considera tais fatores em c_{ijt} (o custo de entrega da fração dos conteúdos solicitados i pelo servidor j no período t) e procura minimizá-lo. Também se procura minimizar a quantidade de ocorrência de *backlogs* de uma requisição i ao longo do tempo b_{it}, já que o custo por *backlog*, p_{it}, é sempre mais alto do que o custo de atendimento, dentro do período, por um servidor mais distante. Finalmente, a função procura minimizar o custo de replicar o conteúdo k no servidor j a partir do servidor l no período t, h_{kjlt}. Mais detalhes a respeito das definições dos custos e do modelo matemático podem ser encontrados em Neves (2011).

5 Experimentos computacionais

Nesta Seção são apresentados os experimentos computacionais envolvendo a heurística POP-PPRDR para avaliar a eficiência do algoritmo proposto, comparando-o com outro método disponível na literatura. Para a definição do tamanho dos subproblemas avaliados e para ajustar o valor do parâmetro que controla o número de iterações da busca local embutida na heurística proposta, conduziu-se experimentos que são relatados em Goldbach (2013).

5.1 Ambiente computacional e instâncias de teste

Todos os experimentos computacionais relatados foram realizados em um equipamento com processador Intel Core i7-2600, 3,40GHz e 16 GB de memória RAM, rodando Linux 3.2.0-52 ubuntu 12.04 LTS 64-bit. Os algoritmos foram implementados na linguagem C++ e compilados com o compilador g++ v. 4.3.

Para avaliar a qualidade da heurística POP-PPRDR buscou-se instâncias sobre as quais houvesse literatura a respeito e que estivessem disponíveis. Para tal, utilizou-se um conjunto de 12 instâncias sintetizadas por Neves (2011). Considerando n o número de servidores, o conjunto

é formado por: duas instâncias com n = 30, identificadas por Instância 13 e Instância 15; cinco instâncias com n = 100, denominadas de Instâncias de 1 a 5; e uma instância para cada valor de n ∈ {200, 300, 400, 500, 600} identificadas por Instância 1. A identificação das instâncias seguiu a mesma nomenclatura usada em Neves (2011). Mais detalhes sobre as instâncias podem ser encontrados em Goldbach (2013).

5.2 Comparação de POP-PPRDR contra o método ORIGINAL

Nesta Seção comparam-se os resultados da aplicação de dois métodos para resolver o PPRDR: o método proposto por Neves (2011), identificado por ORIGINAL e o algoritmo proposto POP-PPRDR, baseado na meta-heurística POPMUSIC e que usa o procedimento ILSAM como método de busca embutido. Todos os testes foram executados no mesmo ambiente computacional, descrito na Seção 5.1. Para cada instância examinada, o algoritmo escolhido foi executado cinco vezes, uma para cada diferente semente do gerador de números aleatórios usado na busca local ILSAM (3, 5, 7, 11 e 13). Os tempos de execução são expressos em segundos.

O Algoritmo ORIGINAL foi executado para as 12 instâncias apresentadas na Seção 5.1 (ao todo, 60 execuções). Conforme Neves (2011), o critério de parada é o tempo. Em cada execução, as iterações se sucederam até que o tempo acumulado desde o início da execução até o final de uma dada iteração igualasse ou ultrapassasse 7.200 segundos. Além desta norma, adotou-se a regra adicional de interromper a execução após 14.400 segundos. A Tabela 1 apresenta resultados médios obtidos nos testes de ORIGINAL e POP-PPRDR. Os resultados estão expressos em milhares. Para cada quantidade de servidores (colunas), a primeira linha apresenta a identificação da instância, e na linha seguinte o valor médio da solução inicial. As quatro linhas seguintes mostram, respectivamente para os algoritmos ORIGINAL e POP-PPRDR, o valor médio da solução e tempo computacional em segundos. Pode-se notar que, para a Instância 1 de tamanho 600, ORIGINAL foi interrompido aos 14.400 segundos sem ter obtido nenhuma melhoria. ORIGINAL também não conseguiu obter melhoria nas execuções para as Instâncias 1 com 200, 300 e 500 servidores, e obteve melhoria apenas discreta para as Instâncias 1 a 5 com 100 servidores. No entanto, para a Instância 1 com 400 servidores, obteve resultado bastante melhor ao obtido por POP-PPRDR, embora em tempo bastante superior aos 7.200 segundos. Em cada coluna, estão assinalados o melhor resultado obtido e o tempo respectivo.

	Quantidade n de servidores						
	30	100	200	300	400	500	600
Instâncias	13 e 15	1 a 5	1	1	1	1	1
Sol. inicial	6.584	65.742	2.001.957	6.091.048	19.717.438	43.029.944	36.113.407
ORIGINAL	6.579	65.741	2.001.957	6.091.048	19.226.447	43.029.944	36.113.407
Tempo (seg.)	7.213	7.277	8.329	8.191	11.623	12.382	14.400
POP-PPRDR	6.569	65.672	2.001.928	6.091.037	19.717.431	43.029.940	36.113.402
Tempo (seg.)	6.895	6.895	7.287	7.214	7.207	7.104	7.318

Tabela 1. Resultados de execução dos algoritmos ORIGINAL e POP-PPRDR. Instâncias e respectivas quantidades de servidores conforme indicado.

As 60 execuções do POP-PPRDR obedeceram ao limite máximo de 40 iterações sem melhora no laço de busca local, com o valor de r definido como 6. O critério de parada foi a inexistência de subproblemas a otimizar. Nos casos em que o tempo de execução foi maior do que 7.200 segundos, foram observados e registrados os resultados obtidos o mais próximo possível de 7.200 segundos.

A Tabela 2 apresenta dados médios obtidos nos testes com as Instâncias 13 e 15 com 30 servidores. O valor médio da solução inicial para as instâncias é 6.584.080. A tabela reporta, para cada algoritmo, a média dos valores obtidos e respectivos tempos computacionais (segunda e terceira colunas) e a média dos melhores valores e respectivos tempos computacionais (quarta e quinta colunas). O POP-PPRDR atingiu um valor médio de 6.568.599 para a função objetivo, 0,24% melhor do que a solução inicial, melhor do que o resultado médio de ORIGINAL (6.578.883, 0,08% melhor do que a solução inicial), e melhor mesmo que o melhor resultado de ORIGINAL (6.577.980). O melhor valor alcançado por POP-PPRDR, 6.567.077, é 0,26% melhor

do que a solução original, uma melhora 2,8 vezes maior do que a obtida por ORIGINAL (0,09%). Os tempos para alcançar as soluções em geral e a melhor solução foram menores em POP-PPRDR do que em ORIGINAL. O consumo de tempo de POP-PPRDR foi de apenas 35,85% do tempo de ORIGINAL. Em termos do melhor valor alcançado, POP-PPRDR consumiu apenas 42,46% do tempo utilizado por ORIGINAL.

Algoritmo	Média dos resultados	Média dos tempos (segundos)	Média do melhor resultado	Média dos tempos do melhor resultado
POP-PPRDR	6.568.599	2.586	6.567.077	2.559
ORIGINAL	6.578.883	7.213	6.577.980	6.027

Tabela 2. Resultados de execução dos algoritmos ORIGINAL e POP-PPRDR. Instâncias 13 e 15 com 30 servidores.

A Figura 2 ilustra o desempenho dos dois algoritmos processando a Instância 13 com 30 servidores. Pode-se notar que POP-PPRDR obtém melhores resultados do que ORIGINAL, e em tempo menor. Nesta figura e em todas as outras que exibem os resultados, o eixo horizontal indica o tempo, expresso em mil segundos.

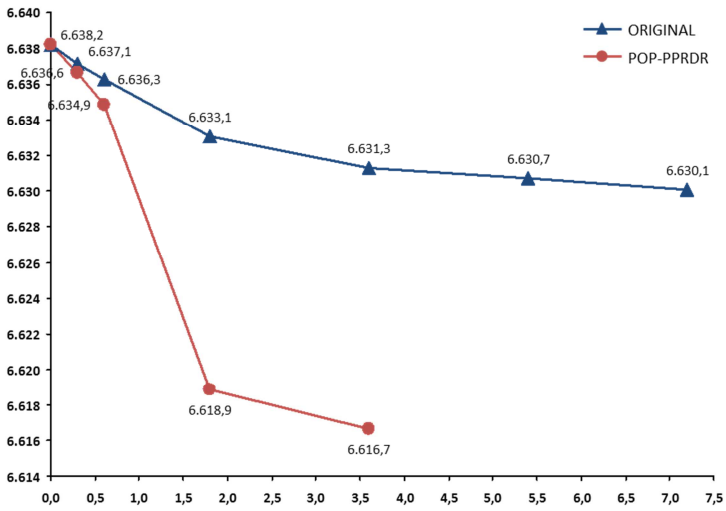

Figura 2. Resultados de execução dos algoritmos ORIGINAL e POP-PPRDR. Instância 13 com 30 servidores. Curvas e eixo Y: valores do custo da solução expressos em milhares.

Considerando os diferentes critérios de parada dos algoritmos, decidiu-se comparar os resultados de POP-PPRDR nas proximidades de marcos de tempo fixos, estabelecidos em ORIGINAL: 300, 600, 1.800, 3.600, 5.400 e 7.200 segundos, registrando a média das execuções com cada uma das cinco sementes. A Tabela 2 apresenta o desempenho consolidado de cada um dos dois algoritmos para as cinco instâncias com 100 servidores e para as cinco sementes do gerador de números aleatórios usado na busca local ILSAM. O valor médio da solução inicial para as instâncias é 65.741.572. As linhas exibem a média dos valores obtidos e respectivos tempos computacionais (segunda e terceira colunas) e a média dos melhores valores e respectivos tempos computacionais (quarta e quinta colunas) para o POP-PPRDR até 7.200 segundos e até finalizar, e para ORIGINAL. O percentual de melhora (0,106%) em relação à solução inicial dos resultados nas proximidades do marco de 7.200 segundos (6.895 segundos na média para o POP-PPRDR), 65.671.863, é 143 vezes maior do que o percentual de melhora (0,000007%) da solução obtida por ORIGINAL (65.643.790), consumindo 95% do tempo utilizado por ORIGINAL. O percentual de melhora do melhor valor alcançado pelo algoritmo POP-PPRDR próximo a 7.200 segundos (6.843 segundos), 65.669.302, foi 0,11%, 48 vezes melhor do que o atingido pelo ORIGINAL (0,000023%, valor de 65.740.058). Para isso, consumiu 1,8 vezes mais tempo. O tempo médio total de execução do POP-PPRDR, até não haver mais subproblemas a otimizar, foi

2,17 vezes maior do que ORIGINAL. Na média, a melhora percentual obtida pelo POP-PPRDR (0,149%, valor de 65.643.790) foi 200 vezes melhor do que melhora a obtida pelo algoritmo ORIGINAL. O melhor resultado obtido por POP-PPRDR, 65.636.748, representa uma melhora de 0,159% em relação à solução inicial, 69 vezes melhor do que o percentual de melhora de ORIGINAL, consumindo para isso 4,65 vezes mais tempo do que ORIGINAL.

Algoritmo	Média dos resultados	Média dos tempos (segundos)	Média do melhor resultado	Média dos tempos do melhor resultado
POP-PPRDR	65.671.863	6.895	65.669.302	6.843
POP-PPRDR ao fim	65.643.790	15.777	65.636.748	17.761
ORIGINAL	65.741.085	7.277	65.740.058	3.822

Tabela 2. Resultados de execução dos algoritmos ORIGINAL e POP-PPRDR. Instâncias 1 a 5 com 100 servidores.

A Figura 3 mostra os resultados obtidos pelos dois algoritmos para a Instância 3 com 100 servidores. É possível observar que o POP-PPRDR obtém melhores resultados do que ORIGINAL e, em tempo menor.

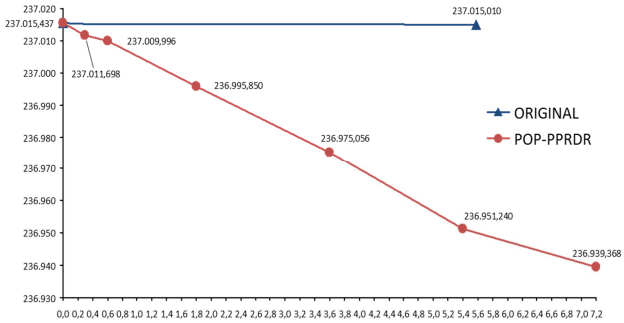

Figura 3. Resultados de ORIGINAL e POP-PPRDR até 7.200 segundos. Instância 3 com 100 servidores. Curvas e eixo Y: valores do custo da solução expressos em milhares.

Para instâncias com mais de 100 servidores, os experimentos de Neves (2011), repetidos no presente trabalho, não obtiveram resultados conclusivos no tempo proposto. A Figura 4 mostra os resultados do POP-PPRDR para a Instância 1 com 200 servidores (esquerda) e Instância 1 com 600 servidores (direita).

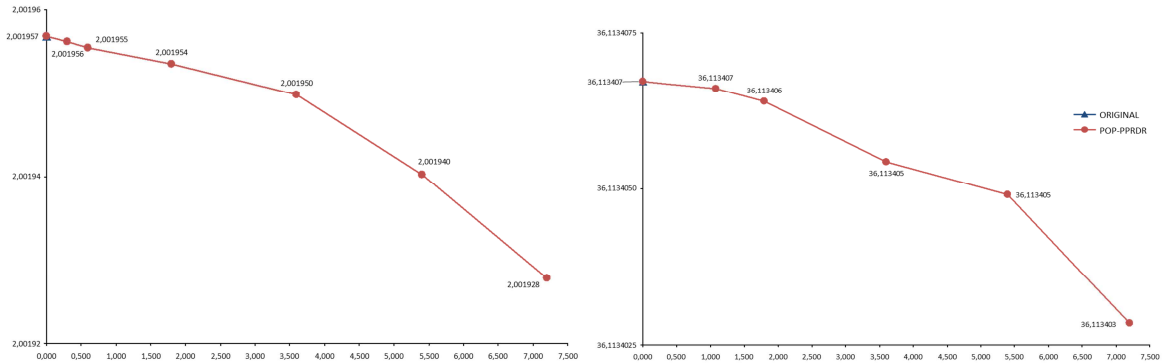

Figura 4. Resultados de POP-PPRDR até 7.200 segundos. Instância 1 com 200 (e) e com 600 (d) servidores. Curvas e eixo Y: valores do custo da solução expressos em bilhões.

A Figura 5 mostra os tempos obtidos pelo POP-PPRDR para tratar as instâncias de tamanho 30, 100, 200, 300, 400, 500 e 600, até a finalização do algoritmo pelo critério proposto por POPMUSIC (inexistência de subproblemas a otimizar).

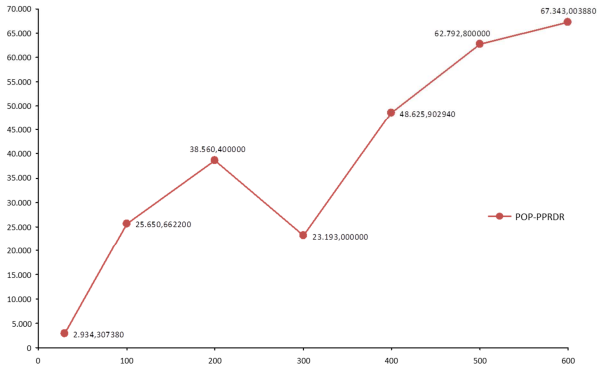

Figura 5. Resultados de POP-PPRDR, até parar, em função do tamanho da instância. Curvas e eixo Y: tempo expresso em segundos; eixo X: quantidade de servidores na instância.

6 Conclusões e trabalhos futuros

Os resultados apresentados demonstraram que a aplicação de POP-PPRDR ao PPRDR consegue, para instâncias de tamanho pequeno a médio, gerar resultados melhores e em tempo computacional menor do que o algoritmo ORIGINAL, proposto por Neves (2011). Para instâncias de tamanho grande, o POP-PPRDR conseguiu iterar e gerar melhoras que não tinham sido possíveis no algoritmo ORIGINAL.

A seguir, relacionam-se algumas propostas de trabalhos futuros com o objetivo de tentar melhorar o algoritmo proposto POP-PPRDR:

Pode-se utilizar a variante mais rápida do método POPMUSIC (veja Seção 2), retirando do conjunto de controle O todas as partes de um subproblema para o qual não se obtenha melhora, no lugar do modelo clássico adotado neste trabalho, no qual se retira de O apenas a parte semente.

No presente trabalho os subproblemas criados pelo método POPMUSIC têm seis partes (r=6). Subproblemas menores exigiram menor tempo de resolução, e subproblemas de tamanho maior exigiram mais tempo. Não se obteve um mesmo valor de r que gerasse o melhor resultado para todos os tamanhos de instância testados. Pode-se ampliar o estudo buscando não um valor fixo mas uma regra, como variar o valor de r em função do número $|V|$ de servidores. Outra possível variação na montagem dos subproblemas poderia incluir a escolha de uma parte com fraco grau de relacionamento com a parte semente, forçando uma diversificação.

Outro estudo que pode merecer atenção é avaliar se haveria correlação entre a disposição geográfica dos servidores em uma RDC e o tamanho dos subproblemas que trouxesse melhores resultados na solução de um problema.

Também é possível estudar a adoção de critérios de parada diferentes do proposto pelo *template* POPMUSIC, por exemplo, em função da quantidade de iterações, que poderia ser relacionada à quantidade de partes da instância. Nos testes, os resultados de POP-PPRDR nas proximidades da 100ª iteração (em instâncias de tamanho 100) apresentaram uma melhora média 113 vezes maior do que a melhora média obtida pelo algoritmo ORIGINAL, consumindo apenas 77,18% do tempo gasto por ORIGINAL.

Como pode ser observado no framework de POPMUSIC (Figura 1, linhas 7-11), sempre quando o subproblema sendo analisado obtém uma melhora, independentemente do seu valor, reabilita todas as partes do subproblema que tenham sido retiradas de O para que possam voltar a servir de semente, prolongando a execução do algoritmo. Na prática, para as instâncias testes analisadas, verificou-se que o valor da melhora às vezes era ínfimo; em alguns casos equivaleram a menos do que 0,5% do valor de outras melhoras para a mesma instância. Uma possibilidade de investigação é estabelecer um valor mínimo de melhora para reabilitar as partes do subproblema sendo considerado, com o objetivo de reduzir o tempo de execução do algoritmo.

Neste trabalho apresentou-se uma aplicação do método de decomposição POPMUSIC para instâncias de médio a grande porte do problema PPRDR. A mesma técnica pode ser estudada em instâncias grandes de outros problemas para os quais seja possível definir, de forma adequada, partes de uma dada solução e estabelecer uma relação de proximidade entre as diferentes partes.

Referências

Almeida, J. M., Eager, D. L., Vernon, M. K. e Wright, S. J. (2004), Minimizing Delivery Cost in Scalable Streaming Content Distribution Systems, *IEEE Transactions on Multimedia*, 6, 356-359.

Alvim, A. C. F. e Taillard, É. D. (2009), POPMUSIC for the point feature label placement problem, *European Journal of Operational Research*, 192, 396-413.

Alvim, A. C. F. e Taillard, É. D. (2013), POPMUSIC for the World Location-Routing Problem, *EURO Journal on Transportation and Logistics*, 2, 231-254.

Bakiras, S. e Loukopoulos, T. (2005), Combining replica placement and caching techniques in content distribution networks, *Computer Communications* 28, 1062-1073.

Bartolini, N., Lo Presti, F. e Petrioli, C. (2003), Optimal Dynamic Replica Placement in Content Delivery Networks. In: *Proceedings of 11th IEEE International Conference on Networks – ICON*, 125-130, Sydney, Australia.

Garey, M. R. e Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, USA, 1979.

Goldbach, R. (2013), *POPMUSIC Aplicada ao Problema de Posicionamento de Réplicas e de Distribuição de Requisições em Redes de Distribuição de Conteúdos*. Dissertação de Mestrado, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, Brasil.

Huang, C. e Abdelzaher, T. (2004), Towards Content Distribution Networks with Latency Guarantees. In: *Proceedings of 12th IEEE International Workshop on Quality of Service – IWQOS*, 181-192, Montreal, Canada.

Lourenço, H. R., Martin, O. e Stützle, T., Iterated Local Search, em: F. Glover and G. Kochenberger, *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, 57, 321-353, Kluwer Academic Publishers, Norwell, MA, USA, 2002.

Neves, T. A., Uchoa Barboza, E., Drummond, L. M. A. e Albuquerque, C. (2008), Otimização em Redes de Distribuição de Conteúdos. *LX Simpósio Brasileiro de Pesquisa Operacional – SBPO*.

Neves, T. A. (2011), *Redes de Distribuição de Conteúdo: Abordagens Exatas, Heurísticas e Híbridas*. Tese de D.Sc., Universidade Federal Fluminense, Niterói, Brasil.

Ostertag, A., Doerner, K. F., Hartl, R. F., Taillard, É. D. e Waelti, P. (2009), Popmusic for a Real World Large Scale Vehicle Routing Problem with Time Windows, *Journal of the Operational Research Society*, 60, 934-943.

Reeves, C. R. e Beasley, J. E., Introduction, em: Reeves, C. R.. (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Introduction chapter, Blackwell Scientific Publications. Oxford, UK, 1993.

Shahabi, C. e Banaei-Kashani, F. (2002), Decentralized Resource Management for Distributed Continuous Media Server, *IEEE Transactions on Parallel and Distributed Systems*, 13, 710-727.

Taillard, É. D. e Voss, S. (2001), POPMUSIC – Partial Optimization Metaheuristic Under Special Intensification Conditions. *Journal of Network and Computer Applications*, 2, 515-540.

Taillard, É. D., Raileanu, L. e Waelti, P. (2007), Preprocessing and Clustering Large-scaled Problem Instances. *Metaheuristic International Conference 07*. Presentation slides. Montreal, Canada.

Tenzakhti, F., Day, K. e Ould-Khaoua, M. (2004), Replication algorithms for the world wide web, *Journal of Systems Architecture*, 50, 591-605.

Wauters, T., Coppens, J., De Turck, F., Dhoedt, B. e Demeester, P. (2006), Replica Placement in Ring-based Content Delivery Networks, *Computer Communications*, 29, 3313-3326. Elsevier.

Zhou, X. e Xu, C.-Z. (2007), Efficient algorithms of video replication and placement on a cluster of streaming servers, *Journal of Network and Computer Applications*, 30, 515-540. Elsevier.