Pré-prova de jornal

O algoritmo genético de chave aleatória tendenciosa de múltiplos pais com relinking de caminho implícito e suas aplicações no mundo real

Carlos E. Andrade, Rodrigo F. Toso, José´ F. Gonçalves,
Mauricio GC Resende

PII: S0377-2217 (19) 30948-8
DOI: https://doi.org/10.1016/j.ejor.2019.11.037
Referência: EOR 16173

Aparecer em: *European Journal of Operational Research*

Data de recebimento: 3 de julho de 2019
Data de aceitação: 18 de novembro de 2019

Por favor citar esta artigo como: Carlos E. Andrade, Rodrigo F. Toso, José´ F. Gonçalves, Mauricio GC Resende, O Algoritmo Genético de Chave Aleatória de Multi-Parent com Implicit Path-Relinking e suas aplicações no mundo real, *European Journal of Operational Research (* 2019), doi: https://doi.org/10.1016/j.ejor.2019.11.037

luzes

- Propomos uma nova hibridização de algoritmo genético de chave aleatória / relink de caminho
- A variante BRKGA usa múltiplos pais durante o processo de acasalamento
- O PR Implícito é independente do problema e opera no espaço de solução BRKGA
- Os testes foram realizados em grandes problemas do mundo real
- Os testes sugerem que a nova abordagem oferece benefícios de desempenho em relação ao padrão BRKGA

O algoritmo genético de chave aleatória com enviesamento de múltiplos pais com relink de caminho implícito e seus aplicações do mundo real

Carlos E. Andrade uma, *, Rodrigo F. Toso b, José F. Gonçalves CD, Mauricio GC Resende d, e

uma AT&T Labs Research, 200 South Laurel Avenue, Middletown, NJ 07748 EUA
b Microsoft AI & Research, 555 110th Ave NE, Bellevue, WA 98004 EUA
c Universidade do Porto, Rua Dr. Roberto Frias, s / n, 4200-464, Porto, Portugal
d Amazon.com, Inc., 399 Fairview Avenue North, Seattle, WA 98109 EUA
e Universidade de Washington, 3900 E Stevens Way NE, Seattle, WA 98195 EUA

Abstrato

Neste artigo, apresentamos o Algoritmo Genético de Chave Aleatória com Tendência Múltipla com Relinculação de Caminho Implícito (BRKGA-MP-IPR), uma variante do Algoritmo Genético de Chave Aleatória Polarizada que emprega pais múltiplos (tendenciosos) para gerar - erate o ff spring em vez dos dois usuais, e é hibridizado com um novo procedimento de busca local de reconexão de caminho implícito. Ao operar sobre o hipercubo de unidade padrão, tal mecanismo de reconexão de caminho alavanca a representação da população do BRKGA e, portanto, fornece independência completa entre o procedimento de busca local e a definição e implementação do problema. Essa abordagem contrasta com os procedimentos tradicionais de reconexão de caminhos que estão vinculados à estrutura do problema. Ter o BRKGA e o IPR operando no mesmo espaço de solução não só torna o paradigma de intensificação / diversificação mais natural, mas também simplifica muito o esforço de desenvolvimento da perspectiva do profissional, já que só é necessário desenvolver um decodificador para mapear os vetores de chave aleatória unitária para o espaço de solução do problema em questão. Além de tais benefícios-chave, extensos experimentos computacionais resolvendo problemas do mundo real, como programação de atualização de software pelo ar, problemas de projeto de rede e leilões combinatórios, mostram que o BRKGA-MP-IPR oferece benefícios de desempenho sobre o BRKGA padrão, bem como o BRKGA com vários pais.

Palavras-chave: Algoritmos genéticos, algoritmos de reconexão de caminhos, heurísticas híbridas

* Autor correspondente.

Endereço de e-mail: cea@research.att.com (Carlos E. Andrade), rofran@microsoft.com (Rodrigo F. Toso), jfgoncal@fep.up.pt (José F. Gonçalves), mgcr@uw.edu (Mauricio GC Resende)

1. Introdução

Algoritmos genéticos [26] tornaram-se uma escolha popular para resolver problemas difíceis de otimização do mundo real em grande escala. Essa metaheurística baseada nos princípios de seleção natural / sobrevivência do mais apto oferece um mecanismo eficiente para buscar um espaço de solução que é exponencialmente grande. Como resultado, os algoritmos genéticos são extremamente populares em vários domínios.

Entre as várias variantes de GAs [28], o Algoritmo Genético de Chave Aleatória Polarizado (BRKGA) tem sido usado com sucesso tanto em vários problemas clássicos de otimização combinatória rígida (ver, por exemplo, [37, 39]), bem como em reais -problemas mundiais, como empacotamento [25], programação [4, 5, 9, 33], leilões combinatórios [10], roteamento de veículos [6, 29], agrupamento [7], projeto de rede complexo [8], colocação de máquinas virtuais em data centers [40] e aprendizado de máquina [15]. BRKGA também foi usado para verificar a viabilidade de instâncias de programas inteiros mistos ming problemas para os quais soluções viáveis são difíceis de encontrar [2]. Elementos do BRKGA apareceram em [12, 18, 22, 23]. No entanto, o método BRKGA-

A ologia foi formalmente introduzida em Gonçalves e Resende [24], que de fi niu um framework claro e apresentou os primeiros resultados como tal.

A principal vantagem do BRKGA sobre outras variantes do algoritmo genético talvez seja a rápida convergência para soluções de alta qualidade. Isso é principalmente apoiado pela adoção de elitismo duplo ao gerar fruto através do acasalamento, a saber: (1) um dos pais é retirado de um conjunto de elite que contém os melhores cromossomos encontrados até agora, e (2) os genes da nascente são herdado do pai da elite com uma probabilidade maior. Do ponto de vista prático, o BRKGA permite uma prototipagem rápida, uma vez que o espaço de solução do BRKGA e o espaço de solução do problema são mantidos separados. Lucena et al. [31] apresentou duas variantes do BRKGA. Na versão de definição de gênero (BRKGA-GD), cada cromossomo tem uma bandeira de bit indicando seu "gênero", e o acasalamento é realizado com um indivíduo com o bit ligado e outro com ele desligado. Na versão multi-pai (BRKGA-MP), mais de dois pais são usados para gerar uma nova nascente. Resultados experimentais sugeriram que o BRKGA-MP pode obter melhores resultados do que o BRKGA-GD. Comportamento semelhante foi relatado em [39], ao comparar os resultados para o BRKGA padrão e o BRKGA-GD aplicado ao problema de roteamento de veículos capacitados com janelas de tempo.

Para robustez, normalmente encontramos estratégias de intensi fi cação para problemas específicos, como busca local ou reconexão de caminhos, sendo usadas em algoritmos metaheurísticos. Neste artigo, enfocamos o path-relinking, que explora o bairro formado no caminho entre duas soluções viáveis e foi proposto pela primeira vez por Glover [21]. A ideia principal de uma busca de reconexão de caminhos é construir uma série de modificações em uma solução base que conduza a uma solução guia, na expectativa de que tais soluções intermediárias sejam de alta qualidade devido à combinação de elementos de ambas as soluções. O principal problema com praticamente todas as soluções de otimização robustas que adotam a reconexão de caminho é que o procedimento é desenvolvido em torno do problema, o que torna a reutilização e modularização do código muito difícil.

Neste artigo, estendemos o trabalho de Lucena et al. [31] fortalecendo o BRKGA-MP com um procedimento Implicit Path-Relinking (IPR). Apresentamos e implementamos estratégias de reconexão de caminhos que operam no vetor de chave aleatória do BRKGA, sendo assim genéricas e modulares; em certo sentido, temos uma estratégia de metaintensificação que pode ser emparelhada com a metaheurística BRKGA como blocos de construção para uma estrutura de otimização robusta. Finalmente, validamos nosso trabalho extensa e completamente usando três problemas complexos do mundo real. No primeiro problema, deseja-se projetar uma rede de backhaul sem fio híbrida complexa que mistura várias tecnologias, como 4G / LTE e Wi-Fi, backhauled usando links de fibra ou microondas [8]. Vários fatores de custo são utilizados neste problema, tais como custo do equipamento, custos de manutenção / aluguel de postes, valetamento de fibra, entre outros. No segundo problema, deve-se agendar atualizações de software pelo ar para uma grande base de dispositivos da Internet das Coisas, especialmente carros conectados [3, 4, 5]. Esse problema tem várias restrições práticas, com centenas de milhares de atualizações que precisam ser agendadas. No terceiro problema, abordamos o problema de determinação do vencedor em leilões combinatórios [10]. Esses leilões são usados para vender direitos como espectro de ondas (bandas) em diferentes mercados. Os resultados mostram que a hibridização de BRKGA e IPR pode melhorar significativamente a qualidade da solução quando comparada às versões padrão. com centenas de milhares de atualizações que precisam ser agendadas. No terceiro problema, abordamos o problema de determinação do vencedor em leilões combinatórios [10]. Esses leilões são usados para vender direitos como espectro de ondas (bandas) em diferentes mercados. Os resultados mostram que a hibridização de BRKGA e IPR pode melhorar significativamente a qualidade da solução quando comparada às versões padrão. com centenas de milhares de atualizações que precisam ser agendadas. No terceiro problema, abordamos o problema de determinação do vencedor em leilões combinatórios [10]. Esses leilões são usados para vender direitos como espectro de ondas (bandas) em diferentes mercados. Os resultados mostram que a hibridização de BRKGA e IPR pode melhorar significativ

A estrutura deste artigo é a seguinte. A seção 2 descreve os procedimentos padrão BRKGA e BRKGA-MP. A seção 3 discute a reconexão de caminhos, os procedimentos de IPR e seus detalhes. A seção 4 aborda as opções de hibridização entre BRKGA e IPR. A seção 5 relata os experimentos computacionais. As observações finais são feitas na Seção 6.

2. Um primer sobre BRKGA e o Multi-Parent BRKGA

O BRKGA é um método elitista que envolve uma população de soluções no hipercubo unitário, ou seja, cada solução é representada por um ponto em $[0, 1]^n$ para uma dada dimensão n. Esta representação foi proposta pela primeira vez em Bean [11] e torna o método independente do problema que ele resolve. Uma função de decodificador $f: [0, 1]^n \rightarrow S$ é necessário mapear indivíduos do espaço BRKGA para o espaço do problema S. Uma vez que todos os operadores evolutivos são aplicados no espaço BRKGA, os operadores personalizados com base na estrutura do problema são desnecessários. Isso permite prototipagem e testes rápidos, reduzindo assim os custos de desenvolvimento. Um aspecto crucial do decodificador é que ele deve ser uma função determinística, ou seja, dado um cromossomo, o decodificador deve sempre gerar a mesma solução. Este pré-requisito garante a reprodutibilidade dos resultados, embora não proíba o uso de sinais aleatórios por decodificadores, digamos para desempate - se um gerador de número pseudoaleatório deve ser usado no decodificador, então ele deve ser gerado por um gene específico no cromossomo.

No BRKGA padrão, a população P é particionado em *elite,* P_e, e *não elite* conjuntos. As soluções do conjunto de elite são as melhores de toda a população de acordo com alguma métrica de desempenho (*aptidão).* O conjunto não elite contém o restantes indivíduos ($P \setminus P_e$). Em cada iteração BRKGA, o conjunto de elite é copiado para a população da próxima geração P' e uma pequena porcentagem de soluções aleatórias

(chamado *mutantes,* P'_m) também é adicionado a esta nova população. Considerando que $|P_e \cup P'_m| \leq |P|$, a população é completada com o ff spring, cada um gerado a partir da combinação (*acasalamento)* de um indivíduo escolhido aleatoriamente da elite conjunto e um indivíduo escolhido aleatoriamente do conjunto não-elite. O acasalamento é feito usando cruzamento uniforme tendencioso, onde um gene é retirado do indivíduo de elite com probabilidade ρ, ou, de outra forma, é tirado do indivíduo não pertencente à elite. Para mais detalhes, o leitor pode consultar [24].

Uma vez que a recém-introduzida dinâmica evolutiva "elitista dupla" é um capacitador-chave por trás do sucesso dos BRKGAs padrão, uma extensão natural é melhorá-la ainda mais desenhando genes de uma combinação de vários indivíduos. Esta variante é chamada de algoritmo genético de chave aleatória com enviesamento de múltiplos pais (BRKGA-MP). Um BRKGA-MP seleciona tantos pais quanto o parâmetro π_t determina. Entre estes, π_e são pais de elite e $\pi_t - \pi_e$ são pais não pertencentes à elite. Cada pai tem uma probabilidade de passar seus alelos para o filho, que é calculado levando em consideração o viés do pai. Pai *tendência* é definido por uma função de polarização de ponderação não crescente pré-determinada Φ: N → R + acima de sua classificação *r.* Usamos as funções de viés propostas por [14]:

- Logarítmico: $\Phi(r) = 1 / \log(r + 1)$;
- Linear: $\Phi(r) = 1 / r$;
- Polinomial (em *d):* $\Phi(r) = r^{-d}$;
- Exponencial: $\Phi(r) = e^{-r}$;
- Constante: $\Phi(r) = 1 / \pi_t$.

Observe que o viés constante nos dá uma distribuição uniforme entre os pais escolhidos.

O Algoritmo 1 mostra o procedimento de crossover completo para BRKGA-MP. Nas linhas 1 e 2, os pais são desenhados e classificados de acordo com sua aptidão. As linhas 3-8 calculam as probabilidades de extrair um gene de cada pai. A função classificação : N → N retorna a posição de um determinado cromossomo na lista classificada de pais *Q;* isso, por sua vez, é passado para a função de polarização Φ. Observe que uma etapa de normalização é necessária para calcular as probabilidades adequadas para cada pai (dado pelo vetor *peso).* As linhas 9–12 realizam o cruzamento. Para cada gene, um pai é escolhido usando o conhecido método "roleta" que usa as probabilidades calculadas nas etapas anteriores, e seu alelo é atribuído ao gene do novo cromossomo. Observe que o procedimento é realizado para cada nova fonte gerada.

Para concluir, para robustez também empregamos outras técnicas utilizadas em diversas variantes de algoritmos genéticos e algoritmos metaheurísticos em geral. Um de eles é desenvolver um conjunto eu_p de p populações de forma independente e trocam seus melhores eu indivíduos depois de cada g gerações. Esta estratégia é conhecida como ilha modelo [41], e geralmente melhora a variabilidade dos indivíduos, acelerando a convergência e evitando ótimos locais. A segunda é reiniciar o algoritmo se ele estiver preso em ótimos locais e não puder melhorar a melhor solução após r

Algoritmo 1: Esquema de crossover BRKGA-MP.

Entrada: População P; função não crescente Φ: N $\rightarrow$ R +; valores $\pi_t \leq |P|$, e $\pi_e \leq \min(\pi_t, |P_e|)$.
Resultado: Um novo cromossomo c.

Amostra uniformemente π_e indivíduos de P_e e $\pi_t - \pi_e$ indivíduos de $P \setminus P_e$.
Deixar Q seja o conjunto desses indivíduos;
Ordenar Q em ordem não crescente de aptidão;
$total_{peso} \leftarrow 0$;
para cada $q \in Q$ *na ordem dada* Faz
    $peso(q) \leftarrow \Phi$ (classificação(q));
    $total_{peso} \leftarrow total_{peso} + peso(q)$;
para cada $q \in Q$ Faz
    $peso(q) \leftarrow peso(q) / total_{peso}$;
Deixar c ser um novo cromossomo vazio;
para $i = 1$ *para* n Faz
    Selecione um indivíduo $q \in Q$ ao acaso, com probabilidades definidas por *peso*;
    $c[i] \leftarrow q[i]$;
Retorna c;

gerações, mantendo apenas a melhor solução geral. No entanto, essa estratégia pode ser muito drástica porque destrói partes bem conservadas do genoma que geralmente aparecem em boas soluções durante o processo de evolução. Para evitar esse modo drástico, Andrade et al. [9] introduziu o *procedimento de agitação* que, em vez de reinicializar totalmente a população inteira, aplica modificações aleatórias aos indivíduos do conjunto de elite e reinicializa os indivíduos não pertencentes à elite. Portanto, garantimos diversidade no conjunto não elitista e preservamos a estrutura de convergência no conjunto elitista abalado.

A Tabela 1 lista os parâmetros de cada variante BRKGA, além de parâmetros adicionais para robustez. Uma desvantagem dos BRKGAs em geral é o grande número de parâmetros a serem definidos; frequentemente, ferramentas de ajuste automático de parâmetros são necessárias para encontrar um bom conjunto de valores de maneira eficaz e eficiente.

3. A reconexão do caminho implícito

Path-Relinking (PR) é uma estratégia de intensificação que explora a vizinhança formada no caminho entre duas soluções viáveis [21]. A reconexão de caminhos foi empregada com sucesso em vários contextos, como roteamento [35], abrangendo [34], atribuição [32], lógica [20], escalonamento [1], etc. Para uma pesquisa abrangente, apontamos o leitor para [38]. A ideia principal de uma busca de reconexão de caminho é construir uma série de modificações em uma solução de base que leva a uma solução guia, sob a expectativa de que tais soluções intermediárias serão de alta qualidade devido à combinação de elementos de ambos soluções. Os procedimentos de reconexão de caminho padrão começam escolhendo uma base e uma solução de guia que são suficientes

Tabela 1: Parâmetros BRKGA. A primeira coluna descreve o nome do parâmetro e a segunda sua definição. Na terceira coluna, os parâmetros podem ser aplicáveis a "ambas" variantes, apenas o padrão ("std.") Ou apenas o multi-pai ("MP"). "Extra" indica um parâmetro que endurece a metaheurística.

Parâmetro	Descrição	Versão
$\|P\|$	Tamanho da população	Ambos
$\|P_e\|$	Tamanho do conjunto de elite	Ambos
$\|P_m\|$	Tamanho do conjunto mutante	Ambos
ρ	Probabilidade de que uma nascente herde o alelo de seu pai de elite	Apenas std.
π_t	Número de pais no cruzamento	Apenas MP
π_e	Número de pais de elite no crossover	Apenas MP
Φ	Função de polarização	Apenas MP
p	Número de populações independentes	Extra
eu	Número de indivíduos na troca	Extra
g	Número de gerações para troca	Extra
r	Número de itera- ções que não melhoraram até a reinicialização	Extra

diverso. Então, iterativamente, um componente da solução base é escolhido, removido, substituído por outro componente da solução guia e avaliado. Quando todos os componentes da solução base são substituídos pelos da solução guia, a busca é concluída e a melhor solução intermediária é retornada. Os componentes podem ser arestas e nós de um gráfico (para problemas de gráfico, como cobertura e roteamento), slots em uma programação ou horário, etc.

A única coisa em comum com a maioria das soluções de otimização robustas que adotam a reconexão de caminhos é que o procedimento é desenvolvido em torno do problema, o que torna a reutilização e modularização do código muito difícil. Partimos da abordagem padrão e, doravante, alavancamos a separação entre o espaço do problema e o espaço da solução do BRKGA para propor um *Revinculação de caminho implícito (* Estratégia IPR). Este novo algoritmo constrói o caminho inteiramente no hipercubo unitário do BRKGA, fazendo uso da função decodificadora (já existente) para mapear os vetores intermediários $v \in [0, 1]^n$ no espaço de solução para avaliação. As vantagens dessa abordagem são duplas, pois oferece (1) redução considerável no tempo de pesquisa e desenvolvimento; e (2) hibridização robusta para qualquer algoritmo genético de chave aleatória, como o BRKGA.

Enquanto as implementações tradicionais de RP "entendem" a estrutura do problema operando explicitamente na vizinhança da solução (nós ou bordas do gráfico, slots ou itens do cronograma, etc.), o IPR só pode contar com o implícito

estrutura do vetor de chave aleatória *v.* Em geral, chaves aleatórias v_{eu} são usados das seguintes maneiras:

1. Limites: essas chaves estão vinculadas a variáveis de decisão, como nós, arestas, etc. e geralmente indicam sua inclusão ou exclusão da solução. Por exemplo, Resende et al. [37] propor e analisar um BRKGA para o Problema de cobertura tripla de Steiner onde $v_{eu} \geq 0{,}5$ indica que o conjunto P_{eu} faz parte da cobertura; esta implementação encontrou (até agora) as melhores coberturas para as duas maiores instâncias em um conjunto de problemas padrão;
2. Permutações: O vetor de chave aleatória v pode induzir uma permutação de decisão variáveis primeiro atribuindo a cada variável um índice do vetor e, em seguida, classificando o vetor para uma permutação. Por exemplo, vamos $v = \{0{,}3, 0{,}1, 0{,}6, 0{,}4\}$; e associe a chave 0.3 com o índice 0, a chave 0.1 com o índice 1, etc. Finalmente, classifique v para obter a permutação { 1, 0, 3, 2}. O problema do caixeiro viajante e suas variantes podem ser facilmente codificados dessa forma; os nós são atribuídos sequencialmente às chaves, que são então classificadas para obter uma sequência de cidades a serem visitadas.

Em alguns casos híbridos, v é dividido em sub-vetores de modo que cada sub-vetor seja interpretado de uma maneira particular. Por exemplo, uma representação pode ter v dividido em vários sub-vetores ou segmentos não sobrepostos, cada um com base em permutações ou limites.

Diante dessas opções, propomos diferentes rotinas para acomodar cada uma especificamente. Descrevemos isso nas próximas seções.

3.1. Revinculação de caminho implícita direta

Na estratégia de religação de caminho direto, o algoritmo assume que nenhuma estrutura particular existe em vetores *v,* assim, as chaves são trocadas diretamente entre o titular e as soluções de guia. Embora tal estratégia seja semelhante aos procedimentos de reconexão de caminho padrão, é muito lenta se apenas uma única chave for substituída por vez - requer $O(n^2)$ chama o decodificador, pois para cada chave na solução de base, o procedimento deve encontrar a melhor chave de substituição na solução do guia. Essa rota leva a uma metaheurística que se concentra muito na intensificação, deixando pouco tempo para a diversificação.

Também observamos que uma estrutura oculta deve aparecer nos vetores v ao longo do processo evolutivo, embora nesta seção assumamos um vetor unitário sem qualquer estrutura explícita. Isso pode ser visualizado, por exemplo, no contexto de um problema de caixeiro-viajante, onde cidades próximas formam grupos de chaves semelhantes no vetor de chave aleatória. Generalizando, certas porções de v podem se tornar blocos de construção essenciais para soluções de alta qualidade; conseqüentemente, rasgar tais blocos ao combinar com outra solução pode ser ineficaz, uma vez que a operação pode destruir os blocos.

Portanto, propomos uma abordagem de religação de caminho direta que troca um bloco de chaves (em vez de uma única chave) em cada iteração. O Algoritmo 2 descreve tal procedimento para um problema de minimização. As linhas 1–4 configuram o algoritmo. Primeiro, ele calcula o número de blocos com base no tamanho de bloco fornecido e o

Algoritmo 2: Revinculação de caminhos implícita direta.

Entrada: Vetores *base, guia* $\in [0, 1]^n$; tamanho do bloco *bs*; porcentagem do caminho $P\% \in (0, 1]$; tempo máximo.

Resultado: A melhor solução encontrada como uma tupla (vetor, valor).

```
1  numblocks ← d n / bs e;
2  tamanho do caminho ← d numblocks × P% e;
3  Deixar RB = {1,. . . , numblocks} ser o conjunto de blocos restantes a serem trocados;
4  melhor solução ← ( base, ∞);
5  enquanto RB não está vazio Faz
6      bestblock ← (-1, ∞);
7      para cada quadra ∈ RB Faz
8          fora de jogo ← quadra × bs
9          b velho ← base [o ff conjunto: o ff conjunto + bs];
10         b novo ← guia [o ff set: o ff set + bs];
11         E se não afetaSolução ( b velho , b novo) então
12             RB ← RB \ {bloco};
13             Vá para a linha 7;
14         base [o ff set: o ff set + bs] ← b novo ;
15         valor da solução ← decodificar( base);
16         base [o ff set: o ff set + bs] ← b velho ;
17         E se valor da solução <bestblock.value então
18             bestblock ← ( bloco, valor da solução);
19     fora de jogo ← bestblock. quadra × bs; base [o ff set: o ff set + bs] ← guia
20     [o ff set: o ff set + bs];
21     E se bestblock.value <bestsolution.value então
22         melhor solução ← ( base, bestblock.value);
23     RB ← RB \ {bestblock.block};
24     trocar( base, guia);
25     tamanho do caminho ← tamanho do caminho - 1;
26     E se pathize = 0 ou o tempo máximo é atingido então
27         Vá para a linha 28;
28 Retorna melhor solução;
```

tamanho do caminho a seguir. Posteriormente, esse atributo serve como um critério de parada adicional, permitindo ao usuário controlar a extensão da pesquisa. Em seguida, definimos *RB* como o conjunto a partir do qual os índices de bloco são amostrados. Configuramos a solução base e guia para os vetores de entrada e definimos a melhor solução atual.

O loop principal consiste nas linhas 5–27. Em cada iteração, o algoritmo varre todas as soluções possíveis atribuindo blocos da solução guia para a solução incumbente. Isso é feito no loop interno 7–18. Para cada bloco, calculamos a posição inicial e o deslocamento, e extraímos os blocos correspondentes da solução incumbente e guia.

Na linha 11, comparamos blocos b_1 e b_2 usando a função afetaSolução: $[0, 1]^m \times [0, 1]^m \rightarrow \{verdadeiro\ falso\}$ que retorna verdadeiro se a troca entre b_{velho} e b_{novo} pode produzir uma nova solução. Se esta afirmação falhar, removemos o bloqueio de consideração. Comentaremos sobre esta função mais tarde. Nas linhas 14-15, o o algoritmo copia o bloco da solução guia para a solução incumbente e chama a função decodificadora para avaliar sua adequação. Observe que podemos usar a mesma função de decodificador que usamos no BRKGA. Então, nas linhas 17–18, mantemos o bloco que obteve a melhor solução na iteração atual.

Uma vez que a busca de bloco é finalizada, o algoritmo compromete a melhor mudança na solução incumbente e lembra o melhor valor de solução encontrado até agora (linhas 19–22). O bloqueio é removido de outras considerações e trocamos os papéis da base e das soluções de guia (linhas 23–24); isso implementa o relink bidirecional, que produz soluções melhores do que apenas estratégias de revinculação de caminho para frente ou para trás [36]. As linhas 25-27 adicionam dois critérios de parada além do número de blocos: o tamanho máximo do caminho a ser explorado e o tempo máximo para a pesquisa. Finalmente, o algoritmo retorna a melhor solução encontrada na linha 28.

Fazemos algumas observações sobre o Algoritmo 2. Primeiro, a função afetaSolução desempenha um papel importante em manter o RP longe de explorar regiões já visitadas. Por exemplo, suponha que o vetor v representa limiares para binarização ção onde chave eu está ativo se $v_{eu} > 0,5$. Então, observe que o bloco $b' = [0,1, 0,6, 0,3]$ gera a mesma solução do bloco $b'' = [0,4, 0,9, 0,5]$, e, portanto, não há precisa explorar b''. Função afetaSolução pode ser uma função genérica ou uma função personalizada do usuário adaptada ao problema a ser resolvido. Essa função pode ser usada para modelar muitos tipos de relacionamentos entre os subvetores, como discretizações e permutações de chave pequenas / locais, por exemplo. Além disso, observe que afetaSolução tem um papel importante em fornecer conhecimento ao procedimento de reconexão de caminhos, de forma que não fique totalmente inconsciente da estrutura do problema.

O procedimento de decodificação também é uma parte sensível do IPR direto. Várias implementações do BRKGA incluíram procedimentos de busca local como parte do decodificador para melhor qualidade da solução e também convergência mais rápida. Uma vez que tais procedimentos podem ser caros do ponto de vista computacional, eles afetariam gravemente o desempenho do IPR. Outra técnica comumente encontrada em BRKGAs com busca local é fazer com que o decodificador atualize o cromossomo (ajuste as chaves cromossômicas) para refletir a solução da busca local, já que tal estratégia também acelera a convergência. No entanto, o ajuste do cromossomo invalidará o IPR e, portanto, deve ser desativado, uma vez que as chamadas para o decodificador podem modificar a solução incumbente e interferir na reconexão do caminho.

3.2. Revinculação de caminho implícita baseada em permutação

Como mencionado anteriormente, é comum encontrar algoritmos de chave aleatória que usam chaves aleatórias para induzir permutações nas estruturas de uma solução, ou seja, a ordem das chaves é usada para construir uma solução. Em tais cenários, a reconexão de caminhos implícita direta descrita na seção anterior pode ser ineficiente devido à natureza da representação da solução na qual existe um número infinito de vetores que representam a mesma solução. Portanto, chave simples

a troca frequentemente não resulta em uma nova solução, mesmo se usarmos blocos de teclas e implementarmos uma função adequada afetaSolução. Para resolver as deficiências mencionadas, propomos a reconexão de caminhos implícita baseada em permutação. Neste método, em vez de trocar as chaves entre as soluções base e guia, usamos a permutação da solução guia para trocar as chaves da solução base de modo que induzam a mesma "subpermutação" na base solução.

Vamos supor que temos duas permutações p e p' representando sequências de elementos de duas soluções distintas para o problema (por exemplo, p e p' podem ser sequências de vértices que representam a solução para um problema do caixeiro viajante). Para uma determinada posição eu nessas sequências, o relinking de caminho implícito baseado em permutação verifica se tal posição eu contém o mesmo elemento em ambas as sequências p e p'. Se esse é o caso, eu é ignorado. Caso contrário, o algoritmo muda as chaves da solução de base para induzir a mesma posição de eu na solução do guia. O algoritmo 3 descreve esse procedimento. As linhas 1–4 realizam a configuração inicial calculando o tamanho do caminho, definindo as soluções base e guia e inicializando a melhor solução encontrada até agora neste procedimento. Linha 5 começa o conjunto de índices a serem verificados. Nas linhas 6 e 7, criamos as tuplas eu_b e eu_g para representar as permutações induzidas por vetores v_1 e v_2. É por isso que eles são classificados em ordem não crescente das chaves da solução base e guia, respectivamente.

No loop principal (linhas 8–28), o algoritmo constrói o caminho usando movimentos de troca entre as posições dos elementos nas permutações. Em cada iteração, o loop interno (linhas 11-20) verifica cada posição eu: E se eu Tem o mesmo elementos em permutações eu_b e eu_g, nós o removemos de consideração e analisamos a próxima posição. Caso contrário, trocamos as chaves da solução de base para que o elemento em posição eu corresponde onde é encontrado na solução do guia (linha 15). Para esclarecer, considere o seguinte exemplo. Suponha que o vetor de solução de base seja $b = [$ 0,1, 0,5, 0,9, 0,3, 0,7] e o vetor de solução guia é $g = [$ 0,8, 0,4, 0,2, 0,6, 0,1], induzindo as permutações $eu_{b=(}$ 1, 4, 2, 5, 3) e $eu_{g=(}$ 5, 3, 2, 4, 1), respectivamente. Observe essa posição $i = 3$ tem o mesmo item (elemento 2) em ambas as permutações e nada deve ser feito. No outro mão, posição $i = 1$ tem elementos diferentes em ambas as permutações, ou seja, $eu_b[1] = 1$ e $eu_g[1] = 5$. Uma vez que nosso objetivo é ter o mesmo elemento em posição eu em ambas as sequências, trocamos a chave na posição $eu_b[1] = 1$, qual é $bi_b[1]] = 0,1$, para o chave na posição $eu_g[1] = 5$, qual é $bi_g[1]] = 0,7$. Tal mudança resulta no vetor [0,7, 0,5, 0,9, 0,3, 0,1] e, conseqüentemente, a nova permutação (5, 4, 2, 1, 3).
Uma vez que temos a nova solução de base, o algoritmo a decodifica e muda as chaves de volta para a solução de base anterior. Esse procedimento é realizado para cada posição e mantemos aquela que gera a melhor solução (linhas 18 a 20). Depois de encontrar o melhor movimento, nas linhas 21-25, o algoritmo confirma a mudança, atualiza a melhor solução, remove a posição eu_{melhor} de uma consideração posterior, e troca os papéis da base e as soluções de guia (para frente / para trás revinculação de caminho). As linhas 26–28 implementam critérios de parada adicionais além do número de elementos verificados, e na linha 29, a melhor solução encontrada é retornada.

Algoritmo 3: Revinculação de caminho implícito baseado em permutação.

Entrada: Vetores $v_1, v_2 \in [0, 1]^n$; porcentagem do caminho $P\% \in [0, 1]$; tempo máximo.
Resultado: A melhor solução encontrada como uma tupla (vetor, valor).

```
1  tamanho do caminho ← d n × P% e;
2  base ← v1;
3  guia ← v2;
4  melhor solução ← ( v1, ∞);
5  Deixar RI = {1,. . . , n} ser o conjunto dos demais índices a serem verificados;
6  Deixar eu_b = (1,. . . , n) e eu_g = (1,. . . , n) ser o conjunto de índices de vetores v1
     e v2 respectivamente;
7  Ordenar eu_b e eu_g de acordo com a ordem não crescente das chaves correspondentes em v1
     e v2;
8  enquanto RI não está vazio Faz
9      eu_melhor ← -1;
10     val_melhor ← ∞;
11     para cada eu ∈ RI Faz
12         E se eu_b[i] = eu_g[eu] então
13             RI ← RI \ {eu};
14             Vá para a linha 11;
15         trocar( base[eu_b[eu]], base[eu_g[eu]]);
16         valor da solução ← decodificar( base);
17         trocar( base[eu_b[eu]], base[eu_g[eu]]);
18         E se valor da solução <val_melhor então
19             eu_melhor ← eu;
20             val_melhor ← valor da solução;
21     trocar( base[eu_b[eu_melhor]], base[eu_g[eu_melhor]]);
22     E se val_melhor < bestsolution.value então
23         melhor solução ← ( base, val_melhor);
24     RI ← RI \ {eu_melhor};
25     trocar( base, guia);
26     tamanho do caminho ← tamanho do caminho - 1;
27     E se pathize = 0 ou o tempo máximo é atingido então
28         Vá para a linha 29;
29 Retorna melhor solução;
```

O algoritmo 3 é muito sensível ao tamanho da entrada, uma vez que torna $O(n^2)$ chamadas para o decodificador. Infelizmente, não podemos usar a estratégia de blocos do Algoritmo 2 o ff-the-shelf, uma vez que as permutações entre os blocos não podem ser traduzidas diretamente entre as soluções base e guia. Assim, o usuário deve controlar este processo usando os critérios de parada adicionais, ou seja, tempo máximo ou tamanho máximo do caminho, caso contrário o BRKGA irá gastar muito tempo na intensificação. Tal como acontece com a reconexão de caminho implícita direta, a função de decodificação

é uma parte sensível e todas as diretrizes feitas na Seção 3.1 devem ser observadas na reconexão de caminho implícita baseada em permutação.

3,3. Seleção de soluções para reconexão de caminhos

Um dos requisitos para uma estratégia de reconexão de caminho bem-sucedida é escolher duas soluções de alta qualidade que sejam diferentes o suficiente para que, ao combinar suas subestruturas, possamos produzir uma solução melhor com uma probabilidade razoável.

Como o IPR é construído na mesma plataforma do BRKGA-MP, aproveitamos o modelo de ilha das populações múltiplas, realizando a reconexão entre cromossomos de elite de diferentes populações de uma forma circular. Por exemplo, suponha que temos três populações. A estrutura realiza três operações de reconexão de caminhos: a primeira entre indivíduos de elite das populações 1 e 2, a segunda entre as populações 2 e 3 e a terceira entre as populações 3 e 1. Quando apenas uma população está evoluída, tanto a base quanto o guia os indivíduos devem ser amostrados no conjunto de elite dessa população.

Ao selecionar indivíduos, propomos duas estratégias de amostragem. No primeiro, simplesmente escolhemos o melhor indivíduo de cada população. Essa estratégia tem a vantagem de sempre usar os indivíduos da mais alta qualidade, com a desvantagem de que tais cromossomos podem ser muito semelhantes e, portanto, restringir a vizinhança de reconexão de caminhos. Na segunda estratégia, amostramos indivíduos de conjuntos de elite para melhorar a probabilidade de ter cromossomos que são diferentes o suficiente para resultar em uma reconexão de caminho bem-sucedida. Como nenhuma das duas estratégias garante que os indivíduos sejam suficientemente diferentes, também usamos uma função de distância entre os indivíduos no hipercubo da unidade como um pré-requisito para executar o IPR. Em contraste,

Ao usar representações baseadas em limiares, podemos calcular a distância de Hamming entre os dois vetores, ou seja, o número de posições onde os vetores diferem. Para calcular essa distância, primeiro aplicamos os limites em cada chave dos vetores de chave aleatória para discretizá-los. Após essa transformação categórica, podemos calcular a distância de edição entre os dois vetores. Por exemplo,
deixar $v_1 = [0{,}1, 0{,}3, 0{,}7]$ e $v_2 = [0{,}5, 0{,}1, 0{,}9]$. Usando o limite de 0,4, temos uma distância de um. Agora, usando um limite de 0,6, a distância é zero. Hamming cálculo de distância é executado em O (*n).*

Ao usar a representação baseada em permutação, podemos usar a distância de classificação de Kendall tau [27], que é uma métrica que calcula o número de desacordos entre pares entre duas listas de classificação e, portanto, pode ser usada para medir a distância entre duas permutações. Dados dois vetores v' e v'', deixar τ' e τ'' ser a ordem (classificação) induzida pela classificação v' e v'' não cada vez mais, respectivamente. A distância entre os tau de Kendall τ' e τ'' é

$$K(\tau', \tau'') = \| \{(i, j) : i < j, (\tau'[i] < \tau'[j] \wedge \tau''[i] > \tau''[j]) \vee (\tau'[i] > \tau'[j] \wedge \tau''[i] < \tau''[j])\} \|.$$

Usando v_1 e v_2 do exemplo anterior, temos $\tau' = (3, 2, 1)$ e $\tau'' = (3, 1, 2)$, o que resulta em uma distância de um. A distância de Kendall tau pode ser computado com um simples $O(n \text{ registro } n)$ algoritmo, mas pode ser reduzido a $O(n \text{ registro } n)$ usando o método apresentado em [16].

Em métodos populacionais como o BRKGA, é comum que a população converta para indivíduos semelhantes ao longo do tempo. Portanto, o IPR pode ter que testar muitos, talvez todos os pares de cromossomos elite antes de prosseguir para o algoritmo de reconexão de caminho. Esta operação pode demorar muito e prejudicar o processo de otimização. Portanto, usamos um parâmetro adicional que limita o número (ou porcentagem) de pares de cromossomos testados antes da reconexão do caminho. Se o IPR atinge esse limite sem encontrar um par de cromossomos adequado para a operação, o algoritmo declara falha, indicando que a população é muito homogênea.

Concluímos esta seção listando os parâmetros de IPR na Tabela 2.

Tabela 2: Parâmetros IPR. A primeira coluna descreve o nome do parâmetro e a segunda tem sua definição.

Parâmetro	Descrição
sel	Seleção individual (elite aleatória ou melhores soluções) Número
cp%	/ porcentagem de pares de cromossomos a serem testados
md	Distância mínima entre os cromossomos
typ	Tipo de redirecionamento de caminho (direto ou baseado em permutação)
bs	Tamanho do bloco
ps%	Porcentagem / tamanho do caminho
max.time	Tempo máximo

4. Hibridizações de BRKGA-MP e IPR

Como os métodos BRKGA-MP e IPR operam no mesmo espaço de soluções (o hipercubo unitário), a hibridização entre os dois métodos é direta, conforme parcialmente descrito na Seção 3.3. No entanto, ainda há dois itens a serem discutidos: como misturar as chamadas entre as etapas evolutivas do BRKGA e a reconexão de caminhos; e como lidar com novas soluções originadas da reconexão de caminhos.

Em alguns métodos, como GRASP [36], procedimentos de reconexão de caminho são usados como um método de intensificação após o algoritmo principal ser interrompido. Neste artigo, propomos duas abordagens diferentes. No primeiro, chamamos o IPR periodicamente, após um determinado número de gerações do BRKGA. A justificativa para essa abordagem é que o DPI pode intensificar a busca de soluções alternativas de alta qualidade, potencialmente acelerando a convergência. Na segunda abordagem, o IPR é chamado apenas quando o BRKGA converge e paralisa por um determinado número de gerações. Assim que obtivermos uma nova solução de IPR, esta solução é inserida no

população (ver critério abaixo), e o BRKGA é retomado; Esperançosamente, esse novo indivíduo levará a melhores soluções no BRKGA. Na Seção 5, descrevemos experimentos com ambas as abordagens.

O segundo e mais crítico problema na hibridização é quando inserir soluções de IPR de volta na população BRKGA. Se a solução IPR for a melhor solução encontrada até agora, podemos adicioná-la à população diretamente. No entanto, é possível que a solução IPR não seja a melhor encontrada até agora. Se tal solução for pior do que a pior solução do conjunto de elite, nós a descartamos; caso contrário, calculamos a distância entre a solução IPR e todas as outras soluções no conjunto elite, usando funções de distância conforme descrito na Seção 3.3. Se a nova solução respeitar uma distância mínima entre todas as soluções elite, descartamos a pior solução do conjunto elite e adicionamos esta solução IPR. Essa estratégia tenta manter uma população diversificada. Outra estratégia comumente usada é substituir a solução mais semelhante de pior custo. Nesse caso,

5. Resultados Experimentais e Discussão

Para avaliar o desempenho do BRKGA-MP-IPR, fazemos uso de três cenários da vida real onde o padrão BRKGA foi previamente aplicado [3, 4, 8, 10]. Em cada caso, os cromossomos são usados de uma maneira diferente para representar a solução, a saber, representação baseada em limiar, baseada em permutação e representação híbrida. Além disso, os decodificadores não possuem procedimentos de busca local, o que nos permite avaliar o desempenho do algoritmo evolutivo sozinho, sem interferência de procedimentos de busca local.

Também testamos o mecanismo implícito de reconexão de caminhos por si só, sem a dinâmica evolutiva do BRKGA. Para isso, amostramos os indivíduos para reconexão de caminhos de toda a população.

Os resultados são representados por rótulos como a seguir: std para o BRKGA original / padrão; mp para BRKGA-MP (sem hibridização IPR); ipr para IPR padrão; parar para o híbrido BRKGA-MP-IPR onde o procedimento IPR é acionado quando a evolução do BRKGA não melhora a melhor solução para um determinado número de gerações; fixo para o BRKGA-MP-IPR onde o procedimento IPR é sempre acionado após um determinado número de gerações. Quando necessário, usamos o su ffi x “ dir ”(Respectivamente,“ por ”) Para indicar explicitamente o uso de redirecionamento de caminho direto (resp., Baseado em permutação).

Para cada cenário, ajustamos cuidadosamente cada algoritmo individualmente usando F-race iterada [13, 30] para que possamos extrair os melhores resultados de cada algoritmo. Tendo ajustado cada algoritmo, realizamos dez execuções independentes em cada instância para cada problema. O critério de parada usado foi um tempo máximo de clock ou um número máximo de iterações sem melhorias, ambos definidos independentemente para cada problema.

Os experimentos computacionais foram realizados em um cluster de máquinas idênticas, cada uma com uma CPU Intel Xeon E5-2650 a 2,0 GHz (12 núcleos / 24 threads) e 128 GB de RAM executando CentOS Linux 6.9. Os algoritmos

testados neste artigo são implementados em C ++ e construídos com GNU GCC 7.2. A estrutura BRKGA-MP-IPR é licenciada sob uma licença semelhante a BSD e está disponível em:

- https://github.com/ceandrade/brkga_mp_ipr_cpp (Versão C ++);
- https://github.com/ceandrade/brkga_mp_ipr_julia (Versão Julia);
- https://github.com/ceandrade/brkga_mp_ipr_python (Versão Python).

5.1. Representação híbrida: problema de design de rede de backhaul sem fio

Começamos testando um problema complexo que usa uma representação cromossômica híbrida para resolver o problema de projeto de rede de backhaul sem fio (WBNDP) descrito inicialmente em [8]. WBNDP é um problema muito complexo do mundo real que visa para construir uma floresta de roteamento para pequenas células LTE, misturando saltos sem fio e de fibra. O problema consiste em um conjunto de pontos de demanda, um conjunto de locais onde pequenas células (ra-

estações base dio) podem ser instaladas (geralmente postes em cruzamentos de ruas) e um conjunto de pontos-raiz para onde as células pequenas podem se conectar por meio de links de fibra ou sem fio. Cada ponto de demanda gera tráfego que pode ser convertido em receita. Esse tráfego deve ser servido por células pequenas (LTE, WiFi ou ambos, cada uma com um raio de serviço) com capacidade limitada de tráfego e número de conexões. As small cells devem ser roteadas por links sem fio com capacidade limitada ou uma fibra se estiver no último salto (perto de um ponto raiz). Temos custos de equipamento (dependendo do tipo de equipamento), custos de abertura de valas de fibra e custos de arrendamento de postes. O objetivo é construir uma floresta de roteamento de forma que a receita líquida (receita bruta menos custo de implantação e operação) seja maximizada ao longo de uma determinada janela de tempo. ^{2} para 8.740 pólos, 420 nós de raiz, 35.750 blocos em uma área de 410 km ^{2}

Em [8], o cromossomo é particionado em cinco seções usando uma mistura de representações de limite e permutação. Embora o decodificador seja um tanto complicado, ele não contém rotinas de busca local. Uma vez que não é simples usar a reconexão de caminhos direta ou baseada em permutação em uma representação cromossômica híbrida, testamos ambas as versões. Pelo mesmo motivo, implementamos uma rotina personalizada para calcular a distância entre dois cromossomos: primeiro, calculamos a soma da distância de Kendall Tau para cada bloco indutor de permutação; a seguir, adicionamos a distância de Hamming entre os blocos de limiar correspondentes (genes de parâmetros de ativação, consulte a Seção 5.1 em [8]).

Os parâmetros ajustados para este problema são descritos na Tabela 3. A primeira seção descreve os parâmetros BRKGA (de acordo com a Tabela 1), seguidos pelos parâmetros IPR (descrição na Tabela 2) e os parâmetros extras (também descritos na Tabela 1). Cada linha mostra os parâmetros de cada algoritmo. Observe que "-" indica "não aplicável". A distância mínima entre os cromossomos é proporcional ao tamanho do cromossomo multiplicado pela coluna " *md.* " Para

fixed-dir e fixo por, parâmetro eu_g indica o intervalo, em número de gerações, que o procedimento de reconexão de caminho é chamado. Para stall-dir e

Tabela 3: Parâmetros do algoritmo ajustado para o WBNDP.

	BRKGA							IPR						Extra			
	$\lvert P \rvert$	P_e%	P_m%	ρ	π_t	π_e	Φ	*sel*	*cp*%	*md*	*typ*	*bs*	*ps*%	eu_p	eu_{eu}	eu_g	*R*
std	500	30	15	0,70	-	-	-	-	-	-	-	-	-	3	2	100	300
mp	1800	24	10	-	3	10	e-r	-	-	-	-	-	-	1	-	-	215
fixed-dir	720	32	12	-	7	10	1 / r	Melhor	12	0,26	Dir.	33		3	1	-	400	410
stall-dir	360	22	11	-	4	6	1 / r	Melhor	-	0,12	Perm.	32	31		1	-	150	500
fixo por	1400	43	14	-	5	10	e-r	Melhor	11	0,23	Perm.	50	70		1	-	275	330
tenda-por	950	13	10	-	7	10	1 / r	Melhor	-	0,08	Perm.	35	15		1	-	260	440
ipr-dir	60	-	-	-	-	-	-	Rand.	100	0,12	Dir.	15	92		-	-	-	300
ipr-per	100	-	-	-	-	-	-	Rand.	100	0,15	Perm.	11	85		-	-	-	290

stall-per, parâmetro eu_g indica o número de gerações sem melhorias na melhor solução antes que a reconexão do caminho deva ser chamada. Observe também que, para uma única população, não há troca individual. Usamos uma hora ou 1.000 iterações sem melhorias como critério de parada, e quatro threads para decodificação, mais de 30 instâncias apresentadas em [8].

A Figura 1 mostra a distribuição do desvio percentual médio sobre a melhor solução encontrada para cada instância. Observe que stall-dir e cavalariço apresentam desvios ligeiramente melhores, embora fixo por segue de perto. As percentagens médias e desvios padrão são 7,20 $\pm$ 5,76, 6,68 $\pm$ 5,88, e 7,63 $\pm$ 4,72, respectivamente. mp e fixed-dir apresentou resultados semelhantes com porcentagem média de 8,73 $\pm$ 5,62 e 9,14 $\pm$ 5,71, respectivamente. O std, ipr-dir, e ipr-per algoritmos ficaram aquém (11,19 $\pm$ 5,62, 45,85 $\pm$ 22,10, e 47,52 $\pm$ 19,19, respectivamente).

Para con fi rmar nossas conclusões, testamos a normalidade dessas distribuições usando o teste de Shapiro-Wilk e aplicamos o teste de soma de postos de Wilcoxon, considerado mais eficaz do que o *t*- teste para distribuições suficientemente longe do normal e para tamanhos de amostra suficientemente grandes [17, 19]. Para todos os testes, assumimos um intervalo de confiança de 95% e usamos o método de Bonferroni para *p*- ajuste de valor. Tabela 4

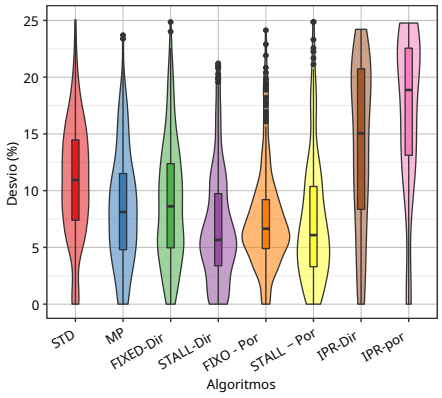

Figura 1: Desvio das melhores soluções encontradas para o WBNDP.

Tabela 4: *p*- valores do teste de soma de postos de Wilcoxon entre os algoritmos do WBNDP. Usamos um intervalo de confiança de 95% e omitimos *p*- valores são menores que 0,05.

	std	mp	fixed-dir	stall-dir	fixed-per	stall-per	ipr-dir
mp	-						
fixed-dir	-	1,00					
stall-dir	-	-	-				
fixo por	-	0,20	-	0,76			
tenda-por	-	-	-	1,00	0,05		
ipr-dir	-	-	-	-	-	-	
ipr-per	-	-	-	-	-	-	1,00

mostra o *p*- valores do teste de Wilcoxon. Também fornecemos, no Apêndice A, o teste de análise de variância (ANOVA) para cada par de algoritmos, embora as distribuições não são normais. Como esperado, os resultados de ANOVA e Wilcoxon diferem um pouco, mas concordam na maioria dos casos. Observe que todas as versões do BRKGA-MP são significativamente melhores do que std e ipr. Não podemos con fi rmar uma diferença estatística entre stall-dir, fixed-per, e stall-per. Também não podemos con fi rmar uma diferença significativa entre mp e fixed-dir, e mp e fixed-dir.

A Tabela 5 relata o desempenho dos algoritmos com relação às melhores soluções encontradas. A coluna "% Sol" mostra a porcentagem do número das melhores soluções encontradas e a coluna "% Run" mostra a porcentagem do número de execuções nas quais o algoritmo encontrou a melhor solução. As duas colunas sob o rótulo "Prop. di ff. " mostram, respectivamente, a média da diferença proporcional entre o melhor valor da solução e o valor alcançado (%), e seu correspondente desvio padrão (σ). Pode-se notar a superioridade do parar versões, com mais melhores soluções encontradas, e um menor desvio das outras soluções.

A Figura 2 mostra os perfis de desempenho de todos os algoritmos. O *x*- eixo mostra o tempo necessário para atingir um valor de solução alvo, enquanto o *y*- eixo mostra a probabilidade cumulativa de alcançar um valor de solução alvo para o tempo determinado no *x*- eixo. Os valores de destino são a melhor solução para cada instância. É interessante

Tabela 5: Desempenho do algoritmo em relação às melhores soluções encontradas no WBNDP.

Alg.	% Sol.	% Corre	Prop. Di ff.	
			%	σ
std	20,00	4,03	11,80	5,24
mp	26,67	5,67	9,39	5,65
fixed-dir	3,33	3,33	9,59	5,62
stall-dir	43,33	7,00	7,89	5,66
fixo por	3,33	2,67	7,98	4,64
tenda-por	40,00	7,00	7,93	5,14
ipr-dir	3,33	1,33	46,56	21,52
ipr-per	3,33	0,67	47,92	18,79

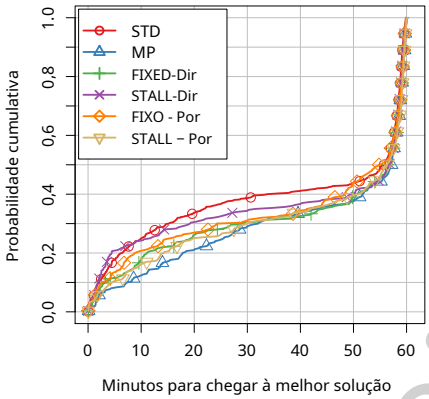

Figura 2: Distribuições de tempo de execução para as melhores soluções encontradas para o WBNDP. As marcas de identificação correspondem a 4% dos pontos traçados para cada algoritmo.

notar que std é mais rápido em encontrar soluções em 50 minutos. No entanto, no final da otimização, todas as variantes exibem um perfil semelhante.

A Tabela 6 traz a quantidade de chamadas do método IPR de acordo com cada variação do algoritmo. A segunda e a terceira colunas mostram o número médio de chamadas (e desvio padrão) por execução e o número máximo de chamadas entre todas as execuções, respectivamente. A quarta e quinta colunas mostram a média e o número máximo de soluções melhoradas obtidas por IPR. As duas últimas colunas mostram o número médio e máximo de vezes que o procedimento de IPR considerou que a população de elite era homogênea e se salvou. Em geral, os métodos IPR foram chamados algumas vezes neste cenário. Observe que stall-dir e fixo por não conseguiu encontrar soluções melhores. De fato, em quase todas as ligações, o IPR declarou a população homogênea demais para reconexão de caminhos. fixed-dir e tenda-por poderia executar IPR algumas vezes, mas apenas uma melhoria foi encontrada em uma determinada execução.

Tabela 6: Estatísticas para chamadas de rotina IPR para WBNDP.

Alg.	Ligações		Improv.		Homog.	
	Avg	Max	Avg	Max	Avg	Max
fixed-dir 2 ± 3	10 0 ± 1		1	0	0	
stall-dir 4 ± 5	21	0	0 4 ± 4		21	
fixo por 3 ± 3	15	0	0 2 ± 3		15	
tenda-por 1 ± 1	5 0 ± 1		1	0	0	

5,2 Representação de permutação: problema de agendamento Firmware-over-the-air

Para testar uma representação direta, usamos o problema de escalonamento Firmware-over-the-air (FOTA), definido em [3, 4]. Neste problema, deve-se criar uma programação para carros conectados para iniciar uma sessão de download / atualização por LTE
redes. No FOTA, são fornecidos dados históricos das conexões do carro e da utilização e capacidade da rede LTE. Com isso, criamos uma capacidade de download
mapa em uma determinada janela de tempo. Carros e restrições de capacidade de rede também são fornecidas. O objetivo é encontrar um cronograma que: 1) mude o download do FOTA para períodos mais silenciosos na rede para evitar horários de pico e, portanto, minimizar o impacto do FOTA na rede; 2) programar tantos carros quanto possível para que eles terminem o download dentro da janela de tempo; e 3) minimizar o tempo total de conclusão do download FOTA. Este problema foi modelado como um problema de programação da máquina. As 153 instâncias variam de 107 trabalhos e 1.676 máquinas a 33.132 trabalhos e 9.916 máquinas em uma janela de tempo de 672 períodos. Devido a várias restrições do mundo real e ao tamanho das instâncias, o problema de agendamento FOTA é extremamente desafiador e difícil de resolver. Andrade et al. [5] apresentou um BRKGA com representação baseada em permutação para escalonar os carros usando um decodificador construtivo simples.

Na Tabela 7, listamos os parâmetros ajustados para este BRKGA baseado em permutação. A distância entre dois cromossomos foi calculada usando o método Kendall tau e é proporcional ao tamanho do cromossomo, conforme descrito anteriormente. Para este cenário, usamos seis horas de relógio de parede ou 1.000 iterações sem melhorias como critério de parada e quatro threads para decodificação. Existem 154 instâncias, conforme descrito em [5].

Na Figura 3, pode-se observar a distribuição do desvio percentual médio em relação à melhor solução encontrada para cada instância. Observe que mp e as variações MP-IPR apresentaram os menores desvios das melhores soluções. Na média, mp shows 1,33 ± 1,34% de desvio, fixo shows 1,16 ± 0,79%, e parar presentes 1,20 ± 0,88%. A versão original, std, resultou em um desvio de 4,19 ± 1,75%, e ipr sozinho resultou em 17,56 ± 8,40%. Aplicando o teste de soma de postos de Wilcoxon, con fi rmamos que todas essas diferenças (par a par) são estatisticamente signi fi cativas em um intervalo de confiança de 95%, exceto para fixo e parar, para o qual o *p*- o valor é maior que 0,05. O Apêndice B traz o teste de análise de variância (ANOVA) para cada par de algoritmos, cujos resultados
concordo com os resultados do Wilcoxon. Portanto, os procedimentos MP-IPR são capazes de encontrar resultados muito melhores do que as variantes BRKGA padrão e multi-pai

Tabela 7: Parâmetros do algoritmo para o problema FOTA.

	BRKGA							IPR							Extra	
	$\lvert P \rvert$	P_e%	P_m%	ρ	π_t	π_e	Φ	*sel*	*cp*%	*md*	*typ*	*bs*	*ps*%	eu_p	eu	eu_g	*R*
std	500	30	15	0,70	-	-	-	-	-	-	-	-	-	-	3	2	100 300
mp	300	26	10	-	2	3	$\frac{1}{\text{registro}(r+1)}$	-	90	-	-	-	-	-	1	-	75 500
fixo	320	24	12	-	4	6	1 / *r*	Rand. 85 0,29 Perm. 19 25 melhores							1	-	75 380
parar	360	22	11	-	4	6	1 / *r*	93 0,12 Perm. 32 31							1	-	65 340
ipr	100	-	-	-	-	-	-	Rand. 100 0,10 Perm. 13 17							-	-	- 240

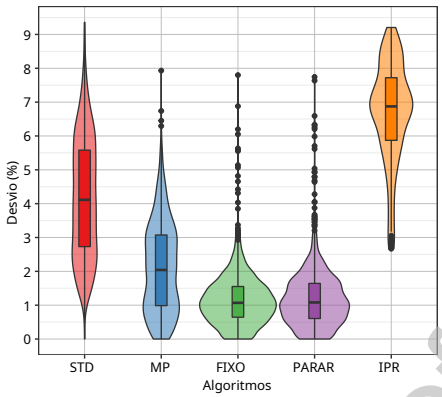

Figura 3: Desvio das melhores soluções encontradas para o FOTA.

sozinho. Isso pode ser con fi rmado na Tabela 8, pois o número das melhores soluções encontradas pelas versões MP-IPR é significativamente maior do que pelos outros algoritmos.

A Figura 4 mostra os perfis de desempenho, semelhantes à Figura 2. O IPR padrão está ausente da figura porque não resultou em uma curva. Observe que todos os algoritmos demoram muito para convergir devido à dificuldade desse problema. std e mp são mais lentos do que os outros algoritmos. Observe que fixo apresentou melhor probabilidade cumulativa do que parar, embora eles convergem para resultados semelhantes no final de suas execuções.

A Tabela 9 expõe a quantidade de chamadas do método IPR de acordo com cada algoritmo. A descrição é semelhante à Tabela 6, mas omitimos as duas últimas colunas, uma vez que não foi encontrada homogeneidade neste cenário. Como esperado, em fixo o IPR é chamado mais vezes, em média, devido à sua natureza cíclica, resultando em mais melhorias. No entanto, mesmo se esperarmos para chamar IPR apenas quando o mecanismo BRKGA parar, podemos melhorar a solução existente; na verdade, parar chama IPR em apenas 20% das execuções, mas essas chamadas ainda são valiosas nos resultados finais.

Tabela 8: Desempenho do algoritmo em relação às melhores soluções encontradas no FOTA.

Alg.	% Sol.	% Corre	Prop. Di ff.	
			%	σ
std	0,65	0,07	4,19	1,75
mp	22,88	2,29	2,16	1,31
fixo	37,91	3,79	1,21	0,77
parar	38,56	3,86	1,24	0,86
ipr	0,00	0,00	17,57	8,40

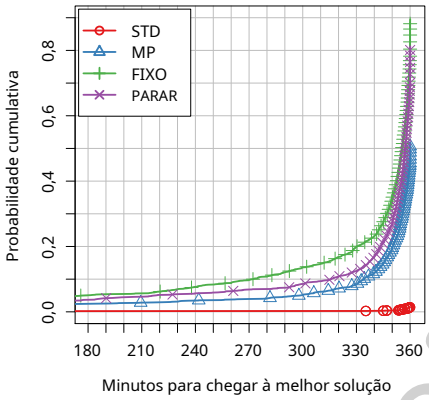

Figura 4: Distribuições de tempo de execução para melhores soluções encontradas no FOTA. As marcas de identificação correspondem a 4% dos pontos traçados para cada algoritmo.

Tabela 9: Estatísticas para chamadas de rotina IPR para FOTA.

Alg.	Ligações		Improv.	
	Avg	Max	Avg	Max
fixo 7 ± 10	64 3 ± 3			16
parar 4 ± 6	44 1 ± 1			4

5.3. Representação de limite: problema de determinação do vencedor

Para alguns problemas, a hibridização entre os métodos BRKGA-MP e IPR pode não ser tão eficaz quanto o esperado. Para mostrar isso, experimentamos o Problema de Determinação do Vencedor (WDP) em leilões combinatórios, em que um vendedor deve escolher um conjunto de lances não sobrepostos para maximizar o valor total de venda. Esses lances são feitos em conjuntos de itens diferentes e possivelmente sobrepostos. Mais formalmente, vamos eu ser um conjunto de itens, $eu_1, eu_2 \in eu$ ser dois itens, e deixe $v: 2^{eu} \rightarrow R$ ser uma função de avaliação para conjuntos desses itens. Itens eu_1 e eu_2 dizem ser *complementar* se e apenas se $v(\{i_1\}) + v(\{i_2\}) \leq v(\{i_1, eu_2\})$. Onde $\{eu_1, eu_2\}$ descreve um pacote de itens eu_1 e eu_2 Diz-se que são *substitutos* se e apenas se $v(\{i_1\}) + v(\{i_2\}) \geq v(\{i_1, eu_2\})$. Cada licitante define suas ofertas para vários subconjuntos de itens de venda, ou seja, cada licitante b enviar uma função v_b para o vendedor. Usando essas funções, o vendedor escolhe as melhores ofertas não sobrepostas, maximizando sua receita. As instâncias para WDP variam de 40 lances a 10 bens a 1.500 lances e 1.500 mercadorias.

Andrade et al. [10] apresentou seis variantes de decodificadores para resolver tal problema, tanto com base em limiar quanto com base em permutação. Aqui, usamos o GA_{RA} variação baseada em limite, onde os lances são classificados por custo / benefício e o decodificador escolhe o lances não sobrepostos, de modo que a chave correspondente no cromossomo seja maior ou igual a 0,5.

Tabela 10: Parâmetros do algoritmo para o WDP.

	BRKGA							IPR							Extra	
	$\|P\|$	P_e%	P_m%	ρ	π_t	π_e	Φ	*sel*	*cp*%	*md*	*typ*	*bs*	*ps*%	eu_p	eu_{eu}	eu_g	*R*
std	2000	20	15	0,70	-	-	-	-	-	-	-	-	-	-	-	3	2 100 500
mp	4600	13	23	-	2	3	r_{-2}	-	92	-	-	-	-	-	-	3	1 60 300
fixo	4670	10	29	-	2	3	e_{-r}	Melhor	97	0,20	Dir.	132	48			3	2 450 470
parar	4500	11	35	-	2	3	e_{-r}	Rand.	94	0,15	Dir.	74			22	3	2 200 470
ipr	70	-	-	-	-	-	-	Rand.	100	0,02	Dir.	5			70	-	- - 400

Os parâmetros ajustados empregados em toda esta seção estão listados na Tabela 10. Usamos uma hora de relógio de parede ou 1.000 iterações sem melhoria como critério de parada e quatro threads para decodificação. Existem 105 instâncias disponíveis para este problema, conforme apresentado em [10].

A Figura 5 mostra a distribuição do desvio percentual médio da melhor solução encontrada para cada instância. Notar que mp, fixo, e parar todos têm pequenos desvios da melhor solução (média e desvio padrão de 0,59 ± 0,75, 0,68 ± 0,75, 0,70 ± 0,90, respectivamente). Observe também que, para as variantes MP-IPR, a maioria das soluções encontradas está bastante próxima das melhores soluções. Aplicando o teste de classificação de Wilcoxon, podemos afirmar que mp, fixo e parar não são significativamente diferentes. No entanto, eles são significativamente melhor que std e ipr. A análise de variância (ANOVA) no Apêndice C mostra resultados ligeiramente diferentes, onde a maioria dos algoritmos tem resultados diferentes em pares. No entanto, como essas distribuições não são normais, esses resultados podem ser distorcidos. Além disso, a Tabela 11 mostra que mp encontraram a porção mais substancial das melhores soluções quando comparadas com as outras variações, também oferecendo a menor diferença proporcional.

Investigamos o uso de IPR neste cenário na Tabela 12. Observe que nenhuma das variantes de IPR poderia encontrar soluções melhores em suas chamadas. Na verdade, para a maioria dos casos, a população de elite era muito homogênea e impedia o IPR de

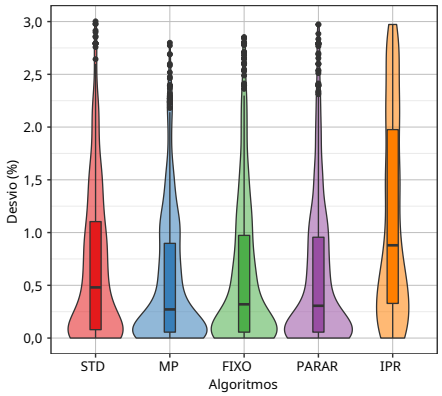

Figura 5: Desvio das melhores soluções encontradas para o WDP.

Tabela 11: Desempenho do algoritmo em comparação com as melhores soluções encontradas para o WDP.

Alg.	% Sol.	% Corre	Prop. Di ff.	
			%	σ
std	43,27	16,44	1.06	1.06
mp	52,88	20,00	0,74	0,78
fixo	50,00	19,90	0,85	0,88
parar	45,19	18,85	0,87	0,92
ipr	4,81	0,67	2,74	2,31

corrida. Na Figura 6, podemos ver que mp tem maior probabilidade de encontrar uma boa solução mais rápido do que as variações MP-IPR, uma vez que mp não perde tempo com chamadas IPR malsucedidas. Pode-se argumentar que também é uma razão para mp para encontrar mais soluções melhores do que as contrapartes de IPR.

Observe que se o profissional tivesse implementado um procedimento customizado de reconexão de caminho para este cenário, o resultado seria muito decepcionante, especialmente depois de investir tempo de desenvolvimento e expectativas apenas para descobrir que um procedimento de reconexão de caminho não é adequado para este cenário. Mesmo que o IPR não pudesse melhorar os resultados, ainda assim economizaria tempo de desenvolvimento e forneceria uma resposta rápida quanto à (in) viabilidade de sua adoção.

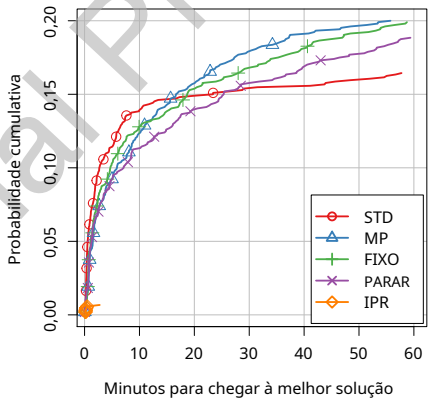

Figura 6: Distribuições de tempo de execução para as melhores soluções encontradas para o WDP. As marcas de identificação correspondem a 4% dos pontos traçados para cada algoritmo.

Tabela 12: Estatísticas para chamadas de rotina IPR para WDP.

Alg.	Ligações		Improv. Homog.			
	Avg	Max	Avg	Max	Avg	Max
fixo 3 ± 1	6		0		$0\,2 \pm 1$	6
parar 5 ± 2	11		0		$0\,5 \pm 2$	11

6. Considerações finais

Neste artigo, apresentamos o algoritmo genético de chave aleatória com polarização múltipla hibridizado com um novo procedimento Implicit Path-Relinking (BRKGA-MP-IPR). O BRKGA-MP usa múltiplos pais para o acasalamento, escolhidos por meio de uma função enviesada para uma convergência mais rápida. O IPR alavanca a representação do hipercubo de unidade genérica do BRKGA e aplica os procedimentos de reconexão de caminho naquele espaço, em vez do espaço do problema, como faz a versão padrão de reconexão de caminho.

Ao testar as variações do BRKGA-MP-IPR extensivamente em problemas do mundo real, concluímos que tanto a dinâmica evolutiva de múltiplos pais quanto o mecanismo de IPR contribuem para encontrar soluções que são muito melhores do que aquelas encontradas pelo BRKGA padrão. Deve-se dizer que o procedimento implícito de reconexão de caminhos é mais sensível à aplicação, a ponto de poder falhar quando a população BRKGA é muito homogênea.

No mínimo, o BRKGA-MP-IPR pode economizar um tempo considerável de desenvolvimento. Usando a mesma infraestrutura, o usuário pode experimentar variantes de BRKGA e IPR incorrendo em sobrecarga de desenvolvimento mínima. Esse fato não só oferece economia para um projeto de otimização usando essas ferramentas, mas também reduz o tempo de colocação no mercado de uma boa solução para o problema subjacente.

Reconhecimentos

Agradecemos aos revisores anônimos por fornecer comentários que aprimoraram este artigo. José F. Gonçalves agradece o apoio do Programa Operacional Regional do Norte de Portugal (NORTE 2020) [número da bolsa NORTE-01-0145- FEDER-000020] no âmbito do Acordo de Parceria PORTUGAL 2020, e através do Fundo Europeu de Desenvolvimento Regional (FEDER) .

Isenção de responsabilidade

O trabalho de Rodrigo F. Toso e Mauricio GC Resende foi feito enquanto eles eram funcionários da AT&T Labs Research em Middletown, NJ, e de José F. Gonçalves quando trabalhava na Universidade do Porto, Portugal.

Referências

[1] RM Aiex, S. Binato, MGC Resende, Paralelo GRASP com caminho-relinking para programação de job shop, Parallel Computing 29 (4) (2003) 393–430, ISSN 0167-8191, doi: 10.1016 / S0167-8191 (03) 00014-0.

[2] CE Andrade, S. Ahmed, GL Nemhauser, Y. Shao, A híbrido primal heurística para encontrar soluções viáveis para programas inteiros mistos, European Journal of Operational Research 263 (1) (2017) 62–71, ISSN 0377-2217, doi: 10.1016 / j.ejor.2017.05.003.

[3] CE Andrade, SD Byers, V. Gopalakrishnan, E. Halepovic, DJ Poole, LK Tran, CT Volinsky, carros conectados em uma rede celular: um estudo de medição, em: Proceedings of the 2017 Internet Mea- surement Conference, IMC '17, ACM, New York, NY, USA, 235-241, doi: 10.1145 / 3131365.3131403, 2017.

[4] CE Andrade, SD Byers, V. Gopalakrishnan, E. Halepovic, DJ Poole, LK Tran, CT Volinsky, Gerenciando grandes atualizações de fi rmware-over-the-air para carros conectados em redes celulares, em: Proceedings of the 2nd ACM International Workshop on Smart, Autonomous and Connected Vehicle System and Services, CarSys ' 17, ACM, ISBN 978-1-4503-5146- 1/17/10, 65–72, doi: 10.1145 / 3131944.3131953, 2017.

[5] CE Andrade, SD Byers, V. Gopalakrishnan, E. Halepovic, DJ Poole, LK Tran, CT Volinsky, Agendamento de atualizações de software para carros conectados com disponibilidade limitada, Applied Soft Computing 82 (2019) 105575, ISSN 1568-4946, doi: 10.1016 / j.asoc.2019.105575.

[6] CE Andrade, FK Miyazawa, MGC Resende, Algoritmo evolucionário para o *k*- Problema de vendedores ambulantes multi-depósito interconectado, em: Processos dos 15◦ Annual Conference on Genetic and Evolutionary Computation, GECCO'13, ACM, New York, NY, USA, ISBN 978-1-4503- 1963-8, 463-470, doi: 10.1145 / 2463372.2463434, 2013.

[7] CE Andrade, MGC Resende, HJ Karlo ff, FK Miyazawa, Evo- algoritmos lucionários para agrupamento de correlação de sobreposição, em: Procedimentos dos 16◦ Conference on Genetic and Evolutionary Computation, GECCO'14, ACM, New York, NY, USA, ISBN 978-1-4503-2662-9, 405–412, doi: 10.1145 / 2576768.2598284, 2014.

[8] CE Andrade, MGC Resende, W. Zhang, RK Sinha, KC Reich- mann, RD Doverspike, FK Miyazawa, Um algoritmo genético de chave aleatória tendenciosa para projeto de rede backhaul sem fio, Applied Soft Computing 33 (2015) 150–169, doi: 10.1016 / j.asoc.2015.04.016.

[9] CE Andrade, T. Silva, LS Pessoa, Minimizing flowtime in a owshop problema de programação com um algoritmo genético de chave aleatória enviesado, Expert Systems with Applications 128 (2019) 67–80, ISSN 0957-4174, doi: 10.1016 / j.eswa.2019.03.007.

[10] CE Andrade, RF Toso, MGC Resende, FK Miyazawa, enviesado algoritmos genéticos de chave aleatória para o problema de determinação de vencedor em leilões combinatórios, Evolutionary Computation 23 (2015) 279–307, doi: 10.1162 / EVCO_a_00138.

[11] JC Bean, algoritmos genéticos e chaves aleatórias para sequenciamento e otimização, ORSA Journal On Computing 2 (6) (1994) 154-160, doi: 10.1287 / ijoc.6.2.154.

[12] NCLF Beirão, Sistema de apoio à decisão para sequenciamento de operações em ambientes Job Shop, URL https: // repositorio-aberto. up.pt/handle/10216/12242, 1997.

[13] M. Birattari, Z. Yuan, P. Balaprakash, T. Stützle, F-Race e iterado F-Race: uma visão geral, em: Métodos experimentais para a análise de algoritmos de otimização, Springer Berlin Heidelberg, 311-336, doi: 10.1007 / 978-3642-02538-9_13, 2010.

[14] JL Bresina, amostragem estocástica com tendência heurística, em: Proceedings of the 13◦ conferência nacional de Inteligência Arti fi cial, vol. 1 de *AAAI'96,* AAAI Press, ISBN 0-262-51091-X, 271-278, 1996.

[15] M. Caserta, T. Reiners, um algoritmo de geração de padrões baseado em pool para análise lógica de dados com ajuste fino automático, European Journal of Operational Research 248 (2) (2016) 593–606, ISSN 0377-2217, doi: 10.1016 / j.ejor.2015.05.078.

[16] TM Chan, M. Pătraşcu, Counting Inversions, O ffl ine Orthogonal Range Counting, and Related Problems, in: Proceedings of the Twenty-First Anual ACM-SIAM Symposium on Discrete Algorithms, ISBN 978-0-89871- 701-3, 978-1-61197-307-5, 161-173, doi: 10.1137 / 1.9781611973075.15, 2010.

[17] WJ Conover, Practical nonparametric statistics, John Wiley & Sons, 2 WL edn., doi: 10.1086 / 407092, 1980.

[18] M. Ericsson, MGC Resende, PM Pardalos, um algoritmo genético para o problema de definição de peso no roteamento OSPF, Journal of Combinatorial Optimization 6 (3) (2002) 299-333, ISSN 1382-6905, doi: 10.1023 / A: 1014852026591.

[19] MP Fay, MA Proschan, Wilcoxon-Mann-Whitney ou teste t? Supondo ções para testes de hipótese e múltiplas interpretações de regras de decisão, Statistics Surveys 4 (2010) 1–39, doi: 10.1214 / 09-SS051.

[20] P. Festa, PM Pardalos, LS Pitsoulis, MGC Resende, GRASP com Path Relinking for the Weighted MAXSAT Problem, Journal of Experimental Algorithmics 11, ISSN 1084-6654, doi: 10.1145 / 1187436.1216581.

[21] F. Glover, Tabu Search and Adaptive Memory Programming - Advances, Aplicações e desafios, cap. 1, Springer US, Boston, MA, ISBN 978-1-4615-4102-8, 1-75, doi: 10.1007 / 978-1-4615-4102-8_1, 1997.

[22] JF Gonçalves, NCLF Beirão, Um algoritmo genético baseado em chave aleatórias para sequenciamento de opearções, Investigação Operacional 19 (1999) 123–137.

[23] JF Gonçalves, JR de Almeida, Um algoritmo genético híbrido para montagem balanceamento de linha, Journal of Heuristics 8 (6) (2002) 629–642, ISSN 1381-1231, doi: 10.1023 / A: 1020377910258.

[24] JF Gonçalves, MGC Resende, Algoritmos genéticos de chave aleatória enviesados para otimização combinatória, Journal of Heuristics 17 (2011) 487-525, ISSN 1381-1231, doi: 10.1007 / s10732-010-9143-1.

[25] JF Gonçalves, MGC Resende, Uma genética multipopulação paralela algoritmo para um problema de empacotamento ortogonal bidimensional restrito, Journal of Combinatorial Optimization 22 (2) (2011) 180–201, ISSN 1382-6905, doi: 10.1007 / s10878-009-9282-1.

[26] JH Holland, Adaptação em sistemas naturais e arti fi ciais: um introdutório análise com aplicações à biologia, controle e inteligência artificial, University of Michigan press, ISBN 0-472-08460-7, 1975.

[27] M. Kendall, Uma nova medida de correlação de classificação, Biometrika 30 (1938) 81–89, doi: 10.2307 / 2332226.

[28] O. Kramer, Genetic Algorithm Essentials, Springer International Publishing AG, ISBN 978-3-319-52156-5, doi: 10.1007 / 978-3-319-52156-5, 2017.

[29] MC Lopes, CE Andrade, TA Queiroz, MGC Resende, FK Miyazawa, Heuristics for a Hub Location-Routing Problem, Networks 68 (1) (2016) 54–90, ISSN 1097-0037, doi: 10.1002 / net.21685.

[30] M. López-Ibáñez, J. Dubois-Lacoste, LP Cáceres, M. Birattari, T. Stützle, O pacote irace: corrida iterada para configuração automática de algoritmo, Operations Research Perspectives 3 (2016) 43–58, ISSN 2214-7160, doi: 10.1016 / j.orp.2016.09.002.

[31] ML Lucena, CE Andrade, MGC Resende, FK Miyazawa, Some extensões de algoritmos genéticos de chave aleatória tendenciosa, em: Proceedings of the 46 ◦ Simpósio Brasileiro de Pesquisa Operacional, XLVI SBPO, 2469-2480, URL http://www.din.uem.br/sbpo/sbpo2014/pdf/arq0357.pdf, 2014.

[32] GR Mateus, MGC Resende, RMA Silva, GRASP com caminho-relinking para o problema de atribuição quadrática generalizada, Journal of Heuristics 17 (2011) 527-567, ISSN 1572-9397, doi: 10.1007 / s10732-010-9144-0.

[33] LS Pessoa, CE Andrade, Heurísticas para um problema de programação de fl uxo-shop com função de objetivo de trabalho gradual, European Journal of Operational Research 266 (3) (2018) 950–962, ISSN 0377-2217, doi: 10.1016 / j.ejor.2017.10. 045.

[34] LS Pessoa, MG Resende, CC Ribeiro, Um híbrido Lagrangean heuristic com GRASP e reconexão de caminho para conjunto *k*- cobrindo, Computers & Operations Research 40 (12) (2013) 3132–3146, ISSN 0305-0548, doi: 10.1016 / j.cor.2011.11.018.

[35] MGC Resende, CC Ribeiro, A GRASP com redirecionamento de caminho para privado roteamento de circuito virtual, Networks 41 (2) (2003) 104-114, doi: 10.1002 / net. 10065.

[36] MGC Resende, CC Ribeiro, Otimização por GRASP, Springer-Verlag New York, ISBN 978-1-4939-6530-4, 978-1-4939-6528-1, doi: 10.1007 / 978-1-4939-6530-4, 2016.

[37] MGC Resende, RF Toso, JF Gonçalves, RMA Silva, A bialgoritmo genético de chave aleatória para o problema de cobertura tripla de Steiner, Optimization Letters (2011) 1–15ISSN 1862-4472, doi: 10.1007 / s11590-011-0285-3.

[38] CC Ribeiro, MGC Resende, Path-relinking intensi? Cation methods for algoritmos de pesquisa local estocástica, Journal of Heuristics 18 (2011) 192-214, ISSN 1572-9397, doi: 10.1007 / s10732-011-9167-1.

[39] AN Rochman, H. Prasetyo, MT Nugroho, Bised random key genetic algoritmo com inserção e seleção de gênero para problema de roteamento de veículos capacitados com janelas de tempo, AIP Conference Proceedings 1855 (1) (2017) 020025, doi: 10.1063 / 1.4985470.

[40] F. Stefanello, V. Aggarwal, LS Buriol, JF Gonçalves, MGC Resende, A Bised Random-key Genetic Algorithm for Placement of Virtual Machines Across Geo-Separated Data Centers, in: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, GECCO '15, ACM, New York, NY, USA, ISBN 978-1-4503-3472-3, 919-926, doi: 10. 1145 / 2739480.2754768, 2015.

[41] D. Whitley, S. Rana, RB Heckendorn, The island model genetic algorítmo: On separability, population size and converence, Journal ofComputing and Information Technology 7 (1998) 33–47.

Apêndice A. Análise de Variância para representação híbrida: Wire-menos problema de design de rede de backhaul

Signif. códigos: 0 '***' 0,001 '**' 0,01 '*' 0,05 '.' 0,1 ' ' 1

>> STD / MP

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	292	292,2	41,01	2,31e-10	***
Resíduos	1018	32080	31,5			

>> STD / FIXED-DIR

	Valor Df	Soma Sq	Média Sq	F	Pr (> F)	
Algoritmo	1	874	874,4	28,17	1,36e-07	***
Resíduos	1018	31599	31,0			

>> STD / STALL-DIR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	2361	2360,9	75,3	<2e-16	***
Resíduos	1018	31917	31,4			

>> STD / FIXED-PER

	Valor Df	Soma Sq	Média Sq	F	Pr (> F)	
Algoritmo	1	1939	1939,3	65,37	1,75e-15	***
Resíduos	1018	30200	29,7			

>> STD / STALL-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	4260	4260	132,3	<2e-16	***
Resíduos	1018	32769	32			

>> STD / IPR-DIR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	256104	256104	1559	<2e-16	***
Resíduos	1018	167183	164			

>> STD / IPR-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	281351	281351	2184	<2e-16	***
Resíduos	1018	131151	129			

>> MP / FIXED-DIR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	29	28,81	0,869	0,352	
Resíduos	598	19824	33,15			

>> MP / STALL-DIR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	113	113,20	3,361	0,0673.	
Resíduos	598	20142	33,68			

>> MP / FIXO-POR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	46	46,36	1.505	0,22	
Resíduos	598	18425	30,81			

>> MP / STALL-PER

	Valor Df	Soma Sq	Média Sq	F	Pr (> F)	
Algoritmo	1	609	609,0	17,35	3,57e-05	***
Resíduos	598	20994	35,1			

>> MP / IPR-DIR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	208094	208094	800,7	<2e-16	***
Resíduos	598	155408	260			

>> MP / IPR-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	227218	227218	1138	<2e-16	***
Resíduos	598	119376	200			

>> FIXED-DIR / STALL-DIR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	256	256,24	7,793	0,00541	**
Resíduos	598	19661	32,88			

>> FIXED-DIR / FIXED-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	148	148,28	4,941	0,0266	*
Resíduos	598	17944	30,01			

>> FIXED-DIR / STALL-PER

	Valor Df	Soma Sq	Média Sq	F	Pr (> F)	
Algoritmo	1	903	902,8	26,32	3,92e-07	***
Resíduos	598	20514	34,3			

>> FIXED-DIR / IPR-DIR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	203226	203226	784,4	<2e-16	***
Resíduos	598	154928	259			

>> FIXED-DIR / IPR-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	222129	222129	1117	<2e-16	***
Resíduos	598	118896	199			

>> STALL-DIR / FIXED-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	15	14,67	0,48	0,488	
Resíduos	598	18262	30,54			

>> STALL-DIR / STALL-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	197	197,11	5,658	0,0177	*
Resíduos	598	20831	34,83			

>> STALL-DIR / IPR-DIR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	217914	217914	839,4	<2e-16	***
Resíduos	598	155245	260			

>> STALL-DIR / IPR-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	237474	237474	1191	<2e-16	***
Resíduos	598	119213	199			

>> FIXO POR / PARADA POR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	319	319,3	9,991	0,00165	**
Resíduos	598	19114	32,0			

>> FIXED-PER / IPR-DIR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	214353	214353	834,9	<2e-16	***
Resíduos	598	153528	257			

>> FIXED-PER / IPR-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	233756	233756	1190	<2e-16	***
Resíduos	598	117496	196			

>> STALL-PER / IPR-DIR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	231219	231219	885,8	<2e-16	***
Resíduos	598	156098	261			

>> STALL-PER / IPR-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	251355	251355	1252	<2e-16	***
Resíduos	598	120065	201			

>> IPR-DIR / IPR-PER

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	420	420,3	0,988	0,321	
Resíduos	598	254480	425,6			

Apêndice B. Análise de Variância para representação de permutação: Problema de programação de firmware over the air

Signif. códigos: 0 '***' 0,001 '**' 0,01 '*' 0,05 '.' 0,1 ' ' 1

>> STD / MP

	Df	Soma Sq	Média Sq	Valor F	Pr (> F)	1
Algoritmo		180948	180948		5051	<2e-16 ***
Resíduos	3028	108477	36			

>> STD / FIXO

	Df	Soma Sq	Média Sq	Valor F	Pr (> F)	1
Algoritmo		203913	203913		5787	<2e-16 ***
Resíduos	3028	106699	35			

>> STD / STALL

	Df	Soma Sq	Média Sq	Valor F	Pr (> F)	1
Algoritmo		203008	203008		5748	<2e-16 ***
Resíduos	3028	106934	35			

>> STD / IPR

	Df	Soma Sq	Média Sq	Valor F	Pr (> F)	1
Algoritmo		134753	134753		3675	<2e-16 ***
Resíduos	3010	110383	37			

>> MP / FIXO

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		693	692,7		576,5	<2e-16 ***
Resíduos	3058	3674	1,2			

>> MP / STALL

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		641	640,6		501,2	<2e-16 ***
Resíduos	3058	3909	1,3			

>> MP / IPR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		3286	3286		1358	<2e-16 ***
Resíduos	3040	7358	2			

>> FIXO / PARADA

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	
Algoritmo	1	1	1.0166		1,459	0,227
Resíduos	3058	2131	0,6967			

>> FIXO / IPR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		6984	6984		3805	<2e-16 ***
Resíduos	3040	5580	2			

>> STALL / IPR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		6817	6817		3564	<2e-16 ***
Resíduos	3040	5814	2			

Apêndice C. Análise de Variância para representação de limite: Win- problema de determinação ner

Signif. códigos: 0 '***' 0,001 '**' 0,01 '*' 0,05 '.' 0,1 ' ' 1

>> STD / MP

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		44,9	44,93	53,98	2,9e-13	***
Resíduos	2078	1729,6	0,83			

>> STD / FIXO

	Df	Soma Sq	Média Sq	F valor	1	Pr (> F)
Algoritmo		21,3	21,296	23,21	1,56e-06	***
Resíduos	2078	1907,0	0,918			

>> STD / STALL

	Df	Soma Sq	Média Sq	F valor	1	Pr (> F)
Algoritmo		16,6	16,620	17,46	3,06e-05	***
Resíduos	2078	1978,4	0,952			

>> STD / IPR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		1759	1759,4	544,2	<2e-16	***
Resíduos	2078	6719	3,2			

>> MP / FIXO

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		4,4	4.360	6,667	0,00989	**
Resíduos	2078	1359,0	0,654			

>> MP / STALL

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		6,9	6,896	10,02	0,00157	**
Resíduos	2078	1430,4	0,688			

>> MP / IPR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		2367	2367	797	<2e-16	***
Resíduos	2078	6171	3			

>> FIXO / PARADA

	Df	Soma Sq	Média Sq	F valor	Pr (> F)
Algoritmo	1	0,3	0,2894	0,374	0,541
Resíduos	2078	1607,8	0,7737		

>> FIXO / IPR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		2168	2167,8	709,6	<2e-16	***
Resíduos	2078	6348	3,1			

>> STALL / IPR

	Df	Soma Sq	Média Sq	F valor	Pr (> F)	1
Algoritmo		2118	2118,0	685,6	<2e-16	***
Resíduos	2078	6419	3,1			