Constraint Programming applied to the Brazilian Book and Teaching Material Program

Lucas Cardoso Silva
Universidade Federal do Paraná (UFPR)
Curitiba, Brazil
lcsilva@inf.ufpr.br

Guilherme A. Derenievicz
Universidade Federal do Paraná (UFPR)
Curitiba, Brazil
guilherme@inf.ufpr.br

Resumo

This paper presents a Constraint Programming model to support the logistics of the Brazilian Book and Teaching Material Program (PNLD). The proposal explores in an integrated manner two global constraints to concisely express the grouping of orders on pallets and the scheduling of production. The experimental results, conducted on the MiniZinc platform with the OR-Tools CP-SAT optimizer, on 34 artificially generated instances, validate the proposed model and indicate that the approach is effective for small-sized instances.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**.

Keywords

Constraint Programming, Logistics, PNLD, MiniZinc

ACM Reference Format:
Lucas Cardoso Silva and Guilherme A. Derenievicz. 2026. Constraint Programming applied to the Brazilian Book and Teaching Material Program. In *Proceedings of Computer on the Beach (CotB '26).* ACM, New York, NY, USA, 7 pages.

1 Introdução

O Programa Nacional do Livro e do Material Didático (PNLD) é um dos pilares da educação pública brasileira, representando uma complexa operação logística para distribuir milhões de materiais didáticos para todo o país dentro de prazos rigorosos [6]. Em Derenievicz et al. [5], os autores analisaram essa logística sob uma perspectiva técnica, propondo um modelo matemático baseado na decomposição do problema em duas etapas: a *Paletização Virtual*, que define a produção e o agendamento da entrega, e a *Roteirização*, que estabelece as rotas de transporte. O estudo destacou como principais desafios a necessidade de acesso a dados reais para validação do modelo proposto e a alta complexidade do problema, caracterizado como um problema de otimização combinatória de grande porte, envolvendo subproblemas de otimização clássicos como o Problema de Carregamento de Paletes (PCP) [10], o Problema de Escalonamento Cumulativo (PEC) [1] e o Problema de Roteamento de Veículos (PRV) [14].

CotB '26, Florianópolis, SC, Brazil

Neste artigo, avança-se o trabalho de Derenievicz et al. [5] ao implementar uma versão simplificada do modelo utilizando Programação por Restrições (PR). A PR é um paradigma para modelar e resolver problemas combinatórios complexos através da declaração explícita das restrições que definem uma solução. Além de tirar proveito de técnicas avançadas de propagação de restrições para acelerar a busca por soluções, a PR mostra-se uma ferramenta versátil e acessível para modelar problemas complexos.

Devido à alta complexidade do PNLD, considerou-se uma simplificação do problema concentrando apenas na produção e na paletização das encomendas. Também foi necessária a relaxação de algumas restrições nestas etapas do processo, visto que tanto o PCP quanto o PEC são problemas NP-Difíceis [1, 10]. Essa simplificação, contudo, segue um critério de preservação estrutural, e não apenas de redução arbitrária do tamanho da instância. O objetivo foi manter explicitamente as decisões que acoplam produção, compatibilidade entre encomendas e consolidação em paletes, enquanto são abstraídos componentes cuja modelagem depende de outra camada logística ou de dados operacionais não disponíveis nesta etapa. Para superar o desafio do acesso dos dados, desenvolveu-se um gerador de instâncias que produz cenários de teste para o modelo proposto. A avaliação experimental demonstrou a capacidade do modelo simplificado em solucionar de forma exata instâncias do PNLD com até 38 encomendas. Embora aquém da dimensão real do PNLD, o uso de métodos exatos permitiu validar o conjunto de restrições proposto. Além disso, para problemas de logística menores, o programa por restrições proposto mostra-se uma alternativa viável de solução.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta o PNLD e o paradigma de PR; a Seção 3 detalha o gerador de instâncias e a formulação simplificada do modelo PR; a Seção 4 apresenta os resultados experimentais; e a Seção 5 conclui o trabalho.

2 Fundamentação

2.1 Programa Nacional do Livro e do Material Didático

O PNLD é uma política pública de grande envergadura do governo brasileiro. Executado pelo Fundo Nacional de Desenvolvimento da Educação (FNDE), tem a missão de garantir o acesso a materiais didáticos de qualidade para todos os estudantes da educação básica pública [6]. A operação logística para cumprir este objetivo é monumental: anualmente, o PNLD realiza cerca de 1,7 milhões de entregas, movimentando 80 mil toneladas de carga, o que se traduz em 280 mil paletes e entre 12 a 17 milhões de encomendas. A estrutura do programa articula uma rede de atores com responsabilidades distintas, incluindo o FNDE como gestor, as editoras

como produtores dos livros, os Correios como o principal operador logístico desde 1994, e as Secretarias Estaduais e Municipais de Educação, que coordenam a distribuição final para as escolas [9].

O estudo de Derenievicz et al. [5] aprofunda a análise técnica do PNLD ao formalizar seus agentes, objetos, restrições e objetivos. Na logística do PNLD, uma vez que as escolas definiram suas demandas de livros e as editoras foram contratadas, inicia-se a etapa denominada *Paletização Virtual*, na qual são definidas as encomendas (agrupamento de exemplares de um item), seus destinatários (escolas), e a sua organização em paletes, além do cronograma de produção dos itens. Esse processo visa otimizar a posterior roteirização e entrega das encomendas.

Na etapa de Postagem, as editoras produzem os itens, montam as encomendas e os paletes, que são então transportados para as centralizadoras do distribuidor. Nas etapas finais, as encomendas são tratadas nas centralizadoras e enviadas em lotes para as escolas. Os seguintes grupos de restrições e função objetivo são identificados por Derenievicz et al. [5] na etapa de Paletização Virtual das encomendas:

- **Paletização:** todas as encomendas devem ser acomodadas em paletes, respeitando os limites físicos de cada unidade. Todas as encomendas em um paletes devem ser do mesmo item, produzidas pelo mesmo produtor e destinadas à mesma centralizadora.
- **Paletes Englobados:** paletes pequenos devem ser englobados em um único palete maior, respeitando os limites físicos do englobamento. Todos os paletes no mesmo englobamento devem ser montados pelo mesmo produtor e destinados à mesma centralizadora.
- **Produção:** cada produtor possui um limite máximo de produção de encomendas de um determinado item por unidade de tempo. Além disso, o processo deve ser organizado para que cada produtor fabrique exemplares de um único item por vez, evitando trocas na linha de produção.
- **Armazenamento e Entregas:** produtores e centralizadoras possuem capacidades máximas de armazenamento. Paletes e encomendas que não são transportadas em um período devem permanecer armazenadas para posterior entrega. Todas as encomendas devem ser entregues até o prazo pré-definido.
- **Função Objetivo:** de modo geral, deseja-se entregar todas as encomendas no menor tempo possível e com o menor custo, o que pode envolver: utilizar a menor quantidade de paletes, efetuar a menor quantidade de entregas e tirar o máximo proveito das rotas.

2.2 Programação por Restrições

A abordagem central da PR envolve a formulação do problema como um Problema de Satisfação de Restrições (PSR). Um PSR é formalmente definido por uma tripla (X, D, C), onde $X = \{x_1, \ldots, x_n\}$ é um conjunto de variáveis, $D = \{D_1, \ldots, D_n\}$ é um conjunto de domínios contendo os valores possíveis para cada variável, e $C = \{c_1, \ldots, c_m\}$ é um conjunto de restrições que especificam as combinações de valores permitidas para subconjuntos de variáveis [4]. A solução de um PSR consiste em encontrar uma atribuição de valores às variáveis que satisfaça todas as restrições simultaneamente. Historicamente, a PR tem raízes na visão computacional [15] e em programação lógica [3].

A resolução do PSR tipicamente combina técnicas de busca (como *backtracking*, busca heurística ou busca local) e propagação de restrições (como a consistência de arco [15]) em um processo de exploração do espaço de busca com podas de sub-espaços que garantidamente não contêm soluções da instância. No caso geral, PSR é um problema NP-difícil [4]. Muitos problemas práticos exigem não apenas uma solução viável, mas a melhor solução de acordo com algum critério. Tais problemas são formulados como Problemas de Otimização com Restrições, que estendem o PSR pela adição de uma função objetivo a ser minimizada ou maximizada [11].

Um elemento crucial para a aplicação prática e a expressividade do paradigma PR é o uso de *linguagens de modelagem de restrições*, que permitem ao usuário declarar o problema de forma concisa e em alto nível. Um poderoso recurso deste paradigma são as *restrições globais*, que encapsulam um conjunto complexo de restrições juntamente com algoritmos de propagação dedicados. Um exemplo clássico é a restrição global `all_different`$(x_1, \ldots, x_n)$, que restringe as n variáveis a assumirem valores distintos entre si. Além da expressividade desta restrição em comparação com outros paradigmas de programação (e.g. Programação Linear Inteira), mecanismos específicos garantem a inferência de valores e redução do espaço de busca de maneira mais eficiente que, por exemplo, considerar as restrições de desigualdade $x_i \neq x_j$ separadamente [12].

Neste trabalho, é utilizada a linguagem de restrições MiniZinc [8], que oferece uma sintaxe de alto nível e uma vasta biblioteca de restrições globais. A linguagem promove a separação clara entre a modelagem do problema e as estratégias de busca empregadas pelos resolvedores subjacentes, como Gecode, OR-Tools, ou Chuffed, com os quais se integra. O Código MiniZinc 1 exemplifica um trecho do modelo proposto neste trabalho. A linha 1 declara um array de variáveis binárias denominado `palete_usado`, indexado pelo conjunto `Paletes`, que indica se um palete foi usado na logística do PNLD ou não (i.e., ficou vazio). Nas linhas 3 a 7, uma restrição garante que todo palete p deve ser assinalado como usado se, e somente se, existir uma encomenda e alocada a esse palete (a variável `palete_encomenda[`e`]` é definida no Código MiniZinc 2). Por fim, a restrição das linhas 9 a 11 garante a quebra de simetria na paletização ao restringir que paletes efetivamente usados apareçam antes dos não utilizados na ordenação, podando ramos da árvore de busca que levariam a soluções equivalentes.

```
1  array[Paletes] of var 0..1: palete_usado;
2
3  constraint forall(p in Paletes)(
4    palete_usado[p] = exists(e in Encomendas)(
5                        palete_encomenda[e] == p
6                      )
7  );
8
9  constraint forall(p in 1..card(Encomendas)-1)(
10   palete_usado[p] >= palete_usado[p+1]
11 );
```

Código MiniZinc 1: Quebra de simetria na paletização.

3 Metodologia

Neste trabalho, os dois principais desafios destacados em Derenievicz et al. [5] são abordados. Primeiramente, um gerador de

instâncias artificiais foi desenvolvido com o objetivo de prover dados e cenários reduzidos para a validação do modelo. Em seguida, a fim de contornar a alta complexidade da logísticia do PNLD, o problema foi analisado e um modelo simplificado foi proposto e implementado na linguagem MiniZinc.

3.1 Gerador de Instâncias Artificiais

O gerador de instâncias desenvolvido recebe como parâmetros as quantidades de produtores (*prod*), centralizadoras (*cent*), itens (*itens*) e paletes (*palet*), além da capacidade dos paletes (*capac.*); e gera como saída uma instância do PNLD descrevendo a quantidade de encomendas, respectivos pesos e destinatários, itens que podem ser produzidos por cada produtor, sua capaciade de produção (em Kg), e o tempo limite para a distribuição. A instância artificial é construída de modo a garantir que uma solução ótima seja conhecida, para fins de validação e comparação dos modelos. Assim, a partir da quantidade pré-definida de paletes, são construídas encomendas a fim de preenchê-los por completo, além de maximizar a produtividade e minimizar o armazenamento dos produtores. O gerador pode ser descrito de acordo com os seguintes passos:

(1) Validar os parâmetros de entrada, garantindo uma instância factível e não-trivial;
(2) Distribuir aleatoriamente os itens aos produtores, de modo que cada item seja produzido por exatamente um produtor;
(3) Gerar aleatoriamente encomendas de pesos variados (dentro de limites mínimo e máximo pé-definidos), até preencher totalmente cada palete;
(4) Distribuir aleatoriamente os paletes preenchidos aos produtores e escolher um item do produtor para atribuir a todas as encomendas do palete;
(5) Definir a capacidade de produção de cada produtor: ao menos um palete de cada item deve ser produzido por período;
(6) Atribuir os paletes às centralizadoras e, destas, aos destinatários.

Um exemplo de instância gerada pelo algoritmo é apresentado na Figura 1, com parâmetros de entrada $prod = 2$, $cent = 2$, $itens = 3$, $palet = 4$ e $capac. = 3$. Nesse cenário, gera-se uma demanda de 2 encomendas aos destinatários 1 e 2, e de 4 encomendas ao destinatário 3. O produtor 1 é capaz de produzir 4 Kg do item 1 e 3 Kg do item 3 por unidade de tempo, enquanto o produtor 2 consegue produzir até 4 Kg do item 2. O gerador também fornece uma solução ótima da instância: no período 1 o produtor 1 produz 3 Kg do item 1 (encomendas 2 e 3) e envia um palete com essas encomendas à centralizadora 2; enquanto o produtor 2 produz 3 Kg do item 2 (encomendas 1 e 8) e envia também um palete à centralizadora 2. No período 2, o produtor 1 produz 3 Kg do item 3 (encomendas 4 e 6) e envia o palete à centralizadora 1; enquanto o produtor 2 produz outros 3 Kg do item 2 (encomendas 5 e 7) e envia também à centralizadora 1. A solução ótima, portanto, demanda o uso de 4 paletes e 2 períodos para a completa produção.

3.2 Relaxação e Simplificação do Modelo

A modelagem matemática apresentada em Derenievicz et al. [5] considera majoritariamente a etapa de Paletização Virtual, sem abordar o PRV associado à entrega dos paletes e encomendas. Mesmo com

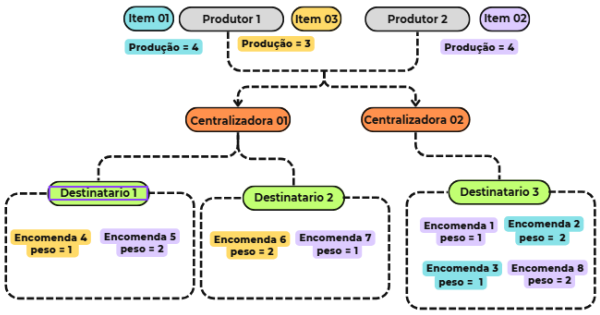

Figura 1: Exemplo de instância artificial produzida pelo gerador.

essa decomposição, o problema formalizado apresenta alta complexidade, com muitas variáveis, restrições e subproblemas NP-difíceis como o PCP e o PEC [1, 10].

Neste trabalho, algumas restrições do modelo original são relaxadas, como as restrições do PCP que são substituídas por restrições do Problema da Mochila. Além disso, a logística como um todo é simplificada para reduzir a dimensão do problema. Considera-se que tais simplificações não descaracterizam a essência do problema e os principais desafios envolvidos. Em particular, o modelo preserva os elementos centrais da etapa de Paletização Virtual que são a homogeneidade dos paletes por produtor, item e centralizadora, capacidade dos paletes,capacidade produtiva por produtor e item em cada período e a restrição de produção de um único item por vez. Em contrapartida, foram abstraídos elementos cuja influência é predominantemente posterior à definição da produção, como a roteirização detalhada até os destinatários, o englobamento residual de paletes e a evolução completa dos estoques. Dessa forma, a formulação continua representando o núcleo combinatório do problema, ao mesmo tempo em que permite validar, de forma controlada, a integração entre paletização e escalonamento da produção.

3.2.1 Demandas e Destinatários. Em Derenievicz et al. [5], é proposto um pré-processamento da instância a fim de agrupar exemplares de itens em encomendas com peso e destinatário fixos. Assim, o conjunto de encomendas torna-se um parâmetro de entrada do algoritmo de otimização. Neste trabalho, um passo adicional é considerado no pré-processamento: as encomendas são assinaladas às centralizadoras dos respectivos destinatários, de modo que a demanda de encomendas passa a ser das centralizadoras. Com esse passo, uma nova decomposição do problema é identificada: (i) entrega de paletes dos produtores às centralizadoras; e (ii) entrega de encomendas das centralizadoras aos destinatários. Considera-se apenas a etapa (i) desta decomposição, sustentado no fato de que a segunda etapa configura uma instância PRV e, portanto, pode ser resolvida por algoritmos específicos [14]. Essa escolha não elimina uma restrição essencial da etapa estudada, ela apenas explicita a fronteira entre a consolidação e expedição dos paletes, tratada aqui, e a distribuição fina até as escolas, deixada para integração futura com métodos específicos de roteamento.

3.2.2 Paletização e Paletes Englobados. Como no PNLD as encomendas são majoritariamente conjuntos de exemplares de livros didáticos e, portanto, homogêneas, entende-se que muitas restrições originais do PCP podem ser relaxadas ou simplificadas. Deste modo, considera-se como restrição apenas que a soma dos pesos das encomendas não exceda a capacidade do palete. Além disso, o conceito de palete englobado não é representado no modelo proposto, uma vez que na prática é aplicado apenas para conjuntos pequenos de encomendas restantes que não completaram paletes inteiros. Nesse contexto, o fator decisório dominante para a etapa analisada passa a ser a compatibilidade logística entre produtor, item e centralizadora, juntamente com o limite agregado de carga do palete, e não a configuração geométrica fina de itens heterogêneos.

O Código MiniZinc 2 apresenta o trecho que modela a paletização das encomendas. A definição de carregamento de paletes se dá pelo array de variáveis `palete_encomenda`, indexado pelo conjunto `Encomendas` e com domínio o conjunto `Paletes`. Assim, a atribuição do valor p à variável `palete_encomenda[`e`]` representa que a encomenda e está sendo carregada no palete p.

```
1  array[Encomendas] of var Paletes:           palete_encomenda;
2  array[Paletes] of var opt Produtores:       produtor_palete;
3  array[Paletes] of var opt Itens:            item_palete;
4  array[Paletes] of var opt Centralizadoras:  centralizadora_palete;
5
6  constraint bin_packing(CapacidadePalete,
7                         palete_encomenda,
8                         peso_encomenda);
9
10 constraint forall(e in Encomendas)(
11   let { var Paletes: p = palete_encomenda[e]; }
12   in
13     produtor_palete[p] == produtor_encomenda[e] /\
14     item_palete[p] == item_encomenda[e] /\
15     centralizadora_palete[p] == centralizadora_encomenda[e]
16 );
```

Código MiniZinc 2: Paletização.

Na linha 6 do código faz-se uso da restrição global `bin_packing`, capaz de representar de forma concisa um conjunto complexo de restrições [8]. Basicamente, essa restrição garante que a atribuição de paletes às encomendas (array de variáveis `palete_encomenda`) seja tal que a soma dos pesos de todas as encomendas em um mesmo palete não exceda a constante `CapacidadePalete`. O peso de cada encomenda é dado pelo parâmetro `peso_encomenda`.

Os arrays de variáveis declarados nas linhas 2 a 4 definem, respectivamente, o produtor, item e centralizadora de cada palete. Conforme as especificações do PNLD, a restrição das linhas 10 a 16 garante que todas as encomendas de um mesmo palete compartilhem do mesmo produtor, item e centralizadora. Nota-se que essas variáveis, na prática, são inferidas diretamente das atribuições às variáveis em `palete_encomenda`. Resolvedores modernos, como o CP-SAT do OR-Tools [2], podem implementar algoritmos de propagação dedicados para problemas como o Bin Packing, melhorando significativamente o desempenho.

3.2.3 Produção e Armazenamento. Neste trabalho, considera-se como unidade mínima de produção um palete completo. Isto é, ao invés de discretizar a produção por exemplares de itens ou por encomendas, conforme proposto por Derenievicz et al. [5], adotou-se a discretização por paletes. Os paletes montados de acordo com as restrições apresentadas no Código MiniZinc 2 contêm encomendas do mesmo produtor e do mesmo item, de modo que essa estratégia respeita a restrição de produção de um único item por unidade de tempo. Essa modelagem reduz a dimensão do problema e não perde aderência ao problema real. As restrições de capacidade máxima de produção (em Kg, por item) continuam sendo satisfeitas. Além disso, o palete passa a ser a unidade natural de interface entre consolidação física, expedição e escalonamento, o que justifica a escolha dessa discretização no modelo simplificado.

No PNLD, encomendas e paletes podem ficar armazenados nos produtores ou centralizadoras a fim de compor cargas que viabilizem rotas de entrega otimizadas. A modelagem apresentada em Derenievicz et al. [5] representa esses cenários através de variáveis de controle de estoque. Neste trabalho, essas restrições foram relaxadas ao considerar a produção por paletes: ao final de cada período, os paletes produzidos podem ser diretamente transportados às centralizadoras. Embora o processo de roteirização das cargas não seja abordado neste trabalho, um critério de qualidade da solução foi incorporado à função objetivo para evitar que uma determinada rota precise visitar muitas centralizadoras no mesmo período (Seção 3.2.4). Vale ressaltar, no entanto, que tal estratégia trata-se de uma simplificação e que uma solução ótima do problema real pode considerar o armazenamento de determinados paletes a fim de compor uma carga mais adequada, composta também por paletes que serão produzidos em períodos posteriores.

O Código MiniZinc 3 apresenta o trecho que modela a produção das encomendas. As variáveis de decisão, no array `inicio_producao`, indicam quando cada palete deve ser produzido. A restrição das linhas 4 a 7 garante que a cada variável `peso_palete[`p`]` é atribuída a soma dos pesos das encomendas carregadas no palete p.

Em seguida, uma restrição global do tipo `cumulative` é definida para cada produtor e cada item que ele pode produzir. De modo geral, essa restrição é da forma `cumulative(inicio_producao, tempo_producao, peso_palete, qtd[q,i])`, garantindo que todas as restrições do PEC sejam satisfeitas. Nesse caso, é atribuído a cada palete p um início de produção, considerando que é necessário `tempo_producao[`p`]` para concluir essa tarefa e que a soma dos pesos de todos os paletes produzidos em um mesmo período não deve exceder a capacidade máxima de produção do item i pelo produtor q.

No entanto, como cada instância de `cumulative` depende do par (q, i), torna-se necessário um conjunto de restrições desse tipo. Assim, para cada produtor q e cada item i (linhas 10 e 11), a restrição deve considerar apenas os paletes produzidos por q e do item i. A linguagem MiniZinc facilita esse processo desconsiderando todos os paletes que têm tempo de produção e peso nulos. Dessa forma, nas linhas 14 a 17 novos arrays são construídos considerando apenas os paletes cujo produtor seja q e item i, zerando os paletes restantes. Nota-se que o tempo de produção de todos os paletes é constante: 1 unidade de tempo.

Por fim, a restrição das linhas 23 a 29 garante que dois paletes do mesmo produtor, mas de itens diferentes, sejam produzidos em períodos distintos, evitando conflitos na capacidade de produção de itens diferentes.

```
array[Paletes] of var Periodos: inicio_producao;
array[Paletes] of var int: peso_palete;

constraint forall(p in Paletes)(
  peso_palete[p] == sum([peso_encomenda[e] | e in Encomendas
                              where palete_encomenda[e] == p])
);

constraint
  forall(q in Produtores)(
    forall(i in Itens where qtd[q,i] > 0)(
      cumulative(
        inicio_producao,
        [if produtor_palete[p] == q /\ item_palete[p] == i then 1
         else 0 endif | p in Paletes],
        [if produtor_palete[p] == q /\ item_palete[p] == i then
         peso_palete[p] else 0 endif | p in Paletes],
        qtd[q,i]
      )
    )
  );

constraint
  forall(p1, p2 in Paletes where
    produtor_palete[p1] == produtor_palete[p2] /\
    item_palete[p1] != item_palete[p2])
    (
      inicio_producao[p1] != inicio_producao[p2]
    );
```

Código MiniZinc 3: Produção.

3.2.4 Função Objetivo. A função objetivo implementada nas linhas 7 a 10 do Código MiniZinc 4 considera três métricas de qualidade de solução, adaptadas do proposto em Derenievicz et al. [5]:

- `ultimo_periodo`: minimizar o tempo total de produção, isto é, o último período em que um palete é produzido. Esse objetivo concede celeridade ao processo de distribuição do PNLD, evitando também o armazenamento prolongado de encomendas nos produtores e centralizadoras;
- `sum(distancia_paletes)`: minimizar a soma das distâncias entre os paletes produzidos em um mesmo período, definida como a quantidade de centralizadoras distintas que precisam ser atendidas pela entrega de paletes em um período. Esse objetivo fornece uma estimativa para o custo da roteirização das cargas;
- `sum(palete_usado)`: minimizar a quantidade de paletes usados. Esse objetivo impacta diretamente na quantidade de entregas necessárias para a conclusão do PNLD.

```
solve :: seq_search([
  int_search(inicio_producao,first_fail,indomain_min,complete),
  int_search(distancia_paletes,first_fail,indomain_min,complete),
  int_search(palete_da_encomenda,first_fail,indomain_min,complete)
])

minimize
  ultimo_periodo*(T_MAX*qtd_Centralizadoras*qtd_Encomendas +
  qtd_Encomendas) + sum(distancia_paletes)*qtd_Encomendas +
  sum(palete_usado);
```

Código MiniZinc 4: Função objetivo.

A formulação da função objetivo segue a lógica de priorização hierárquica, de modo que é preferível minimizar `ultimo_periodo` em detrimento de `sum(distancia_paletes)` e, do mesmo modo, é preferível minimizar este em detrimento de `sum(palete_usado)`. Para garantir essa precedência, a expressão é multiplicada por fatores ponderados de escala elevada. Essa abordagem de objetivo ponderado se alinha com práticas recomendadas em problemas de otimização multiobjetivo, nos quais se busca preservar a prioridade entre critérios por meio da atribuição de pesos lexicográficos [13].

Mais especificamente, o decréscimo de 1 unidade em `sum(distancia paletes)` equivale ao decréscimo de `qtd Encomendas` no valor total da expressão, que é maior que qualquer decréscimo possível em `sum(palete_usado)`, uma vez que no máximo podem ser usados um palete para cada encomenda. A mesma lógica vale a relação entre `ultimo_periodo` e `sum(distancia_paletes)`: com o fator definido nesta linha, qualquer decréscimo na segunda parcela não supera o decréscimo de 1 unidade em `ultimo_periodo`, pois, no máximo, todas as centralizadoras precisam ser visitadas em cada período.

Nas linhas 1 a 5 do Código MiniZinc 4, a anotação `seq_search` define uma estratégia de busca em sequência que explora primeiramente as variáveis do array `inicio_producao`, seguido pelo array `distancia_paletes` e, por fim, `palete_da_encomenda`. Essa ordenação visa explorar primeiro as variáveis que mais impactam na função objetivo, de acordo com a priorização hierárquica definida. Para cada array, aplica-se `int_search` com heurísticas `first_fail` e `indomain_min`, isto é, uma busca sobre domínios finitos que escolhe como próxima variável na árvore de decisão aquela que tem o menor domínio, e como valor o menor do seu domínio. Para as variáveis dos arrays `inicio_producao` e `distancia_paletes` tal heurística explora primeiro as soluções com o menor tempo de distribuição do PNLD e a menor distância entre os paletes, com mais chances de encontrar primeiro as soluções mais próximas do ótima global.

4 Resultados Experimentais

A fim de validar o modelo de PR proposto, foram geradas 34 instâncias artificais com diferentes configurações e executadas na plataforma MiniZinc com o otimizador CP-SAT do OR-Tools [2]. A linguagem de modelagem MiniZinc é atualmente uma das mais adotadas em pesquisas por sua constante atualização e documentação oficial [8], enquanto o solver CP-SAT do OR-Tools é reconhecido pela sua robustez e eficiência em competições de otimização [7].

Os experimentos foram conduzidos em um servidor executando Debian GNU/Linux 12 (Bookworm) com kernel Linux 6.6.6-Atwood, arquitetura x86_64 e processador AMD EPYC 7401 24-Core de 2 GHz, com 256 GB de memória principal. Foi utilizada a versão 2.9.2 do MiniZinc, com o OR-TOOLS CP-SAT 9.12.4544. A invocação do MiniZinc utilizou os parâmetros `-O1`, para ativar o nível 1 de otimização do compilador, e `-f` (*free-search*), que habilita a busca livre pelo otimizador. Para os experimentos, foi definido um tempo limite de 3.600 segundos (1 hora).

Os resultados experimentais estão sumarizados na Tabela 1. Nas colunas 2 a 6 são apresentados os parâmetros utilizados no gerador para cada instância: as quantidades de produtores (*#prod*), centralizadoras (*#cent*), itens (*#itens*) e de paletes (*#palet*), além da capacidade dos paletes (*capac.*). A coluna 7 indica a quantidade total de encomendas (*#encom*), definida automaticamente pelo gerador. As últimas colunas apresentam os resultados obtidos em relação à solução ótima da instância: T é a quantidade de períodos utilizados para a total distribuição (T^* é o valor ótimo); P é a quantidade de paletes utilizados (P^* é o valor ótimo); t_u é o tempo da última solução encontrada pelo otimizador, enquanto t_{total} é o tempo total

Tabela 1: Resultados Experimentais.

id	Parâmetros Gerador						Solução OR-Tools			
	#prod	#cent	#itens	#palet	cap.	#enc	T/T*	P/P*	t_u(s)	t_{tot}(s)
A	2	2	2	2	3	4	1/1	2/2	0,36	0,36
					10	11	1/1	2/2	0,77	0,77
B	3	2	3	3	3	4	1/1	3/3	0,37	0,37
					10	15	1/1	3/3	1,44	1,45
C	2	2	2	4	3	7	1/1	4/4	0,57	0,6
					10	23	2/2	4/4	4,67	TLE
D	3	2	5	5	3	8	2/2	5/5	1,01	1,08
					10	23	2/2	5/5	5,76	9,14
E	3	2	3	6	3	13	1/1	6/6	1,36	1,37
					10	30	2/2	6/6	17,29	TLE
F	3	3	7	7	3	12	3/3	7/7	2,16	168,15
					10	36	3/3	7/7	21,46	TLE
G	3	2	3	10	3	15	3/3	10/10	2,3	242,52
					10	56	2/2	10/10	49,65	TLE
H	3	2	7	7	3	15	3/3	7/7	3,48	445,54
					10	38	3/3	7/7	11,06	1852,47
I	3	2	5	10	3	19	3/3	10/10	5,55	2110,63
					10	58	-/3	-/10	-	TLE
J	3	2	10	10	3	18	4/4	10/10	4,04	TLE
					10	54	4/4	10/10	106,43	TLE
K	3	3	10	10	3	19	4/4	10/10	5,63	TLE
					10	50	-/4	-/10	-	TLE
L	3	2	10	20	3	36	6/6	20/20	28,18	TLE
					10	108	-/6	-/20	-	TLE
M	3	2	7	22	3	38	7/7	23/22	29,06	TLE
					10	124	-/5	-/22	-	TLE
N	3	3	10	20	3	35	6/6	21/20	38,43	TLE
					10	105	-/7	-/20	-	TLE
O	3	3	7	22	3	39	6/6	22/22	109,65	TLE
					10	125	-/7	-/22	-	TLE
P	3	2	10	30	3	56	-/9	-/30	-	TLE
					10	162	-/9	-/30	-	TLE
Q	3	3	10	30	3	60	-/8	-/30	-	TLE
					10	181	-/9	-/30	-	TLE

da execução. Se t_{total} for diferente de TLE, significa que a solução ótima da instância foi encontrada e provada. Caso contrário, não há garantias que a última solução encontrada seja uma solução ótima da instância.

Do total de 34 instâncias, 13 (38,2 %) tiveram prova de otimalidade completa dentro do limite de tempo, 11 (32,4 %) obtiveram ao menos uma solução viável, mas não concluíram a prova de otimalidade, e 10 (29,4 %) não retornaram solução viável. Entre as 11 instâncias sem prova de otimalidade, 9 (82 %) alcançaram o valor ótimo conhecido para ambos os objetivos, ainda que a ótimalidade não tenha sido formalmente comprovada pelo solver. Nas 2 instâncias em que a última solução encontrada não alcançou o ótimo, nota-se que apenas o critério P/P^* não foi atendido (ids M e N, ambos com capac. 3), devido à priorização modelada na função objetivo.

A instância H10 (i.e, id H e capac. 10) foi a maior instância resolvida e provada pelo otimizador dentro do limite de 1h, enquanto as maiores instâncias que tiveram ao menos uma solução viável encontrada foram O3, J10 e G10. Essas 3 instâncias apresentam configurações diferentes, indicando que todos os parâmetros de entrada do gerador interferem na complexidade de solução da instância. Além disso, outras relações não especificadas podem impactar nessa complexidade; por exemplo, H10, que teve a otimalidade provada enquanto outras instâncias menores e maiores, igualmente com capac. 10, atingiram o tempo limite; e C10, que não teve a otimalidade provada.

Nota-se ainda que o desempenho do otimizador reduz com o aumento da capacidade dos paletes de 3 para 10. Essa alteração aumenta a quantidade de encomendas e a complexidade da restrição `bin_packing`, enquanto a complexidade do `cumulative` não sofre alteração, visto que é definida sobre a quantidade de paletes. Dessa forma, conclui-se que ambas restrições impactam diretamente na complexidade do problema.

De modo geral, os experimentos mostram que pequenas variações no número de itens e de produtores, desde que o volume de encomendas seja baixo, têm impacto marginal no custo da busca. Além disso, o otimizador pode rapidamente fornecer soluções ótimas mesmo quando a prova de otimalidade não ocorre dentro do limite de tempo. Isso sugere que, em um contexto prático, poderia-se recorrer a estratégias híbridas, encerrando a busca após a obtenção da primeira solução ótima conhecida ou adotando técnicas de estimativas heurísticas para balancear qualidade versus tempo de execução.

5 Conclusão

Este trabalho avaliou a aplicação do paradigma de Programação por Restrições na logística do PNLD, demonstrando que é possível obter soluções exatas para instâncias de pequeno porte em tempo razoável. Em especial, o método exato utilizado possibilitou validar o modelo proposto. A facilidade de modelagem em MiniZinc e a integração com o OR-Tools mostram que a abordagem escolhida é versátil e acessível para pesquisadores e profissionais de logística. Além disso, problemas logísticos semelhantes ao PNLD, em termos de estrutura ou restrições, podem tirar proveito do modelo aqui proposto.

Para atacar o problema real do PNLD, cuja escala supera o alcance de métodos exatos puros, torna-se necessário combinar esta base de PR com técnicas aproximativas e estratégias de decomposição. Os resultados obtidos neste trabalho indicam que essa combinação deve explorar a complementaridade entre uma fase inicial rápida de construção de solução, uma fase intermediária de reotimização por submodelos menores e uma fase final de intensificação seletiva por Large Neighborhood Search (LNS). Como proposta metodológica inicial, uma heurística construtiva pode gerar rapidamente uma paletização viável e fornecer um limitante superior para o objetivo, em seguida, um submodelo de PR pode reotimizar apenas o escalonamento da produção com os paletes fixados e por fim uma etapa de LNS pode relaxar seletivamente subconjuntos de encomendas ou paletes associados aos produtores, itens ou períodos mais críticos. Como continuidade imediata desta pesquisa, estão em andamento comparações sistemáticas do modelo proposto com heurísticas construtivas, busca local, decomposição em duas fases e abordagens híbridas, visando medir o compromisso entre qualidade, tempo de resposta e robustez.

Acknowledgments

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001. Os autores agradecem ao Kaique Santiago de Paula pelas contribuições no gerador de instâncias artificiais utilizado neste trabalho.

Referências

[1] Abderrahmane Aggoun and Nicolas Beldiceanu. 1993. Extending CHIP in order to solve complex scheduling and placement problems. *Mathematical and Computer Modelling* 17, 7 (1993), 57–73.

[2] B. Altex. 2023. OR-Tools: A Comprehensive Suite for Combinatorial Optimization. *Journal of Optimization Techniques* 15, 2 (2023), 112–145.

[3] Alain Colmerauer and Philippe Roussel. 1996. The birth of prolog. In *History of programming languages-II*. ACM, 331–367.

[4] Rina Dechter. 2003. *Constraint Processing*. Morgan Kaufmann.

[5] Guilherme Derenievicz, Roberto Pereira, Leticia Peres, Marcos Castilho, Nadja Rodrigues, and Silvério da Cruz. 2023. Análise e modelagem da logística do programa nacional do livro e do material didático. *LV Simpósio Brasileiro de Pesquisa Operacional* (6-9 de novembro 2023), 1–14.

[6] FNDE. 2021. *Programa Nacional do Livro e do Material Didático – PNLD: Entendendo a Distribuição*. Technical Report. FNDE. 20p.

[7] MiniZinc Challenge. 2024. Challenge 2024 Results. https://www.minizinc.org/challenge/2024/results/. Acessado em: 2025-06-08.

[8] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. 2007. MiniZinc: Towards a Standard CP Modelling Language. In *Principles and Practice of Constraint Programming 2007, p. 529–543*, Christian Bessière (Ed.). Springer Heidelberg.

[9] Roberto Pereira, Andressa Sebben, Krissia Menezes, Patricia Castellano, Leticia Mara Peres, Guilherme Derenievicz, Marcos Castilho, Nadja Rodrigues, and Silvério da Cruz. 2024. Analyzing the Logistics of the Brazilian Book and Teaching Material Program: a sociotechnical strategy to inform optimization. *Journal on Interactive Systems* 15, 1 (Jul. 2024), 695–711. doi:10.5753/jis.2024.4001

[10] B. Ram. 1992. The pallet loading problem: A survey. *International Journal of Production Economics* 28, 2 (1992), 217–225.

[11] Stuart J. Russell and Peter Norvig. 2021. *Inteligência Artificial: Uma Abordagem Moderna* (4 ed.). GEN/LTC, Rio de Janeiro. Edição Brasileira.

[12] Jean-Charles Régin. 1994. A Filtering Algorithm for Constraints of Difference. *Proceedings of the AAAI Conference on Artificial Intelligence* (1994), 362–367.

[13] Ralph E. Steuer. 1986. *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley & Sons, Inc., New York.

[14] P. Toth and D. Vigo. 2002. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics.

[15] D. Waltz. 1975. Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision*, P. H. Winston (Ed.). McGraw-Hill, 19–91.