

Comparações de modelagens matemáticas para o Coupled-Task Scheduling Problem (CTSP)

André Renato Villela da Silva

Universidade Federal Fluminense

Instituto de Computação

Av. Gal. Milton Tavares de Souza, s/nº

Campus da Praia Vermelha – Boa Viagem – Niterói / RJ

CEP: 24210-346

arvsilva@id.uff.br

RESUMO

O Coupled-Task Scheduling Problem tem o objetivo de alocar N tarefas independentes de forma a minimizar o *makespan*. Cada tarefa i é composta por duas partes a_i e b_i e, entre elas, deve haver um intervalo de tamanho fixo l_i . O primeiro objetivo do artigo é realizar testes empíricos com modelagens matemáticas existentes na literatura e uma nova modelagem de Programação por Restrições, proposta neste trabalho. Os testes envolvem instâncias da literatura. Algumas delas possuem características bem específicas. O segundo objetivo deste trabalho está na verificação do impacto que essas características possuem na execução das modelagens matemáticas. Os resultados mostram que a modelagem proposta foi superior às demais e que existem características que tornam o processo de otimização mais eficiente, em alguns casos, e menos eficiente, em outros.

PALAVRAS CHAVE. CTSP, Programação Inteira Mista, Programação por Restrições.

Otimização Discreta; Outras Aplicações em PO

ABSTRACT

The Coupled-Task Scheduling Problem aims to allocate N independent tasks in order to minimize the problem *makespan*. Each task i is composed of two parts a_i and b_i and, between them, there must be a fixed-size interval l_i . The first objective of the article is to carry out empirical tests with existing mathematical models in the literature and a new Constraint Programming model proposed in this work. The tests involve instances from the literature. Some of them have very specific characteristics. The second objective of this work is to verify the impact that these characteristics have on the execution of the mathematical models. The results show that the proposed model was superior to the others and that there are characteristics that make the optimization process more efficient in some cases and less efficient in others.

KEYWORDS. CTSP, Mixed-Integer Programming, Constraint Programming.

Discrete Optimization; Other applications in OR

1. Introdução

O *Coupled-Task Scheduling Problem (CTSP)* ([Simonin et al., 2011]) é um problema da classe de Problemas de Escalonamento de Tarefas, onde o objetivo é escalonar um conjunto de N tarefas no menor tempo possível (*makespan*). No CTSP, cada tarefa i é composta por duas partes ditas acopladas, respectivamente a_i e b_i , com duração A_i e B_i . Não existe relação de precedência entre duas tarefas do problema, mas após a execução de uma tarefa a_i , deve ser respeitado um intervalo de tempo (*delay*), denotado por l_i , para que a respectiva tarefa b_i possa iniciar sua execução. Não existe a possibilidade de preempção (interrupção parcial das tarefas durante a execução) e não pode haver sobreposição de tarefas.

Nesse artigo, será tratada a versão na qual o *delay* é considerado fixo ([Khatami et al., 2020]), por não haver possibilidade de execução das tarefas b_i nem antes nem depois do tempo indicado pelo intervalo l_i . Como não é possível flexibilizar o início de uma tarefa b_i em relação ao início de sua respectiva tarefa a_i , ao se escalonar uma tarefa, é necessário garantir que suas duas partes não entrem em conflito com outras tarefas já escalonadas. O problema é considerado NP-Difícil ([Shapiro, 1980]).

A Figura 1 mostra um possível escalonamento para três tarefas, onde os tempos da tarefa 1 são $A_1 = 5$, $l_1 = 28$ e $B_1 = 6$. A tarefa 2 é composta por $A_2 = 6$, $l_2 = 7$ e $B_2 = 6$. A tarefa 3 possui os tempos $A_3 = 6$, $l_3 = 6$ e $B_3 = 5$. Como o intervalo da tarefa 1 é significativo, utiliza-se dele para executar outras tarefas e tentar minimizar o *makespan* que, nesse exemplo, é igual a 39 unidades de tempo.

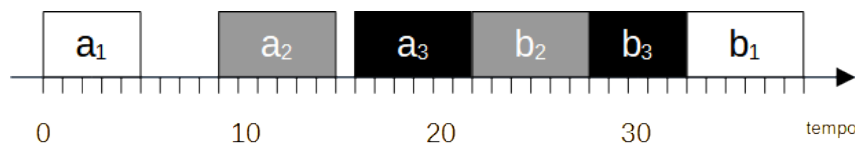


Figura 1: Exemplo de escalonamento com três tarefas

Existem algumas diferentes aplicações para o problema, encontradas na literatura. [Orman et al., 1996] e [Simonin et al., 2011] utilizaram o CTSP em problemas navais. O primeiro artigo, na representação e modelagem de um sistema de radar, onde a tarefa inicial consistia na emissão do pulso de radiofrequência do radar e a segunda tarefa consistia na recepção do pulso refletido de volta. O segundo seguiu em uma linha semelhante, mas modelando o controle de sensores utilizados para guiar torpedos. O CTSP foi utilizado por [Ageev e Kononov, 2007] para representar processos de manufatura química, onde um procedimento inicial deve ter um tempo de espera para que um procedimento final só ocorra quando o produto estiver pronto. Por fim, [Condotta e Shakhlevich, 2014] trataram da marcação de procedimentos médicos de quimioterapia, onde a marcação de duas consultas de um mesmo paciente deve respeitar um período pré-determinado de intervalo entre elas.

Esse trabalho tem dois objetivos principais: propor uma modelagem matemática para o problema na forma de Programação por Restrições, comparando-a empiricamente com outras modelagens matemáticas da literatura, e investigar possíveis características das instâncias do problema que podem tornar a execução dessas modelagens mais ou menos eficientes. Essas são as contribuições teóricas propostas para o avanço da literatura do problema.

O restante desse trabalho está organizado da seguinte forma. Na Seção 2, é feita uma breve revisão da literatura do problema em questão, mostrando algumas variantes do problema e as abordagens mais utilizadas. Na Seção 3, são explicadas a metodologia da pesquisa empregada neste

artigo e as formulações matemáticas que serão testadas. Na Seção 4, são apresentados os resultados computacionais obtidos nos testes realizados. Nas Seções 5 e 6, são trazidas, respectivamente, as principais conclusões obtidas e algumas propostas de trabalhos futuros.

2. Revisão da Literatura

A literatura do CTSP reporta algumas variantes em relação ao problema original. Uma primeira variante considera que os intervalos l_i entre as tarefas a_i e b_i não são fixos, mas limitados entre um valor mínimo e máximo, conforme mostram [Dell'Amico, 1996] e [Potts e Whitehead, 2007]. Na variante “flow-shop”, existem duas máquinas disponíveis para executar as tarefas, uma delas específica para as tarefas a_i e outra específica para as tarefas b_i ([Allahverdi, 2016]).

Existe ainda uma variante onde o objetivo não é minimizar o *makespan*, mas sim executar tarefas, cada uma com seu peso, dentro de um horizonte de tempo limitado, de forma que seja maximizado o peso total das tarefas executadas ([Elshafei et al., 2004]). Por fim, existem variantes onde são definidas precedências entre as tarefas ([Blazewicz et al., 2010]), onde há um escalonamento cíclico ([Brauner et al., 2009]) e onde há incompatibilidade entre as tarefas, ou seja, onde as partes a_i e b_i de uma tarefa não podem ser executadas durante o intervalo l_i de outras ([Simonin et al., 2013]).

A literatura do CTSP apresenta abordagens heurísticas e meta-heurísticas ([Blazewicz et al., 2012], [Blazewicz et al., 2010], [Hwang e Lin, 2004], [Li e Zhao, 2007] e [Condotta e Shakhlevich, 2012]). Foram propostas também algumas modelagens matemáticas na forma de Programação Inteira Mista, como as apresentadas em ([Elshafei et al., 2004], [Sherali e Smith, 2004], [Békési et al., 2014] e [Arabameri e Salmasi, 2004]).

Há diversos algoritmos de complexidade polinomial elaborados para situações bem específicas, onde as tarefas de uma instância apresentam uma característica peculiar. Nesses casos, foi possível elaborar o cálculo de um fator aproximativo como, por exemplo, um algoritmo 1.5-aproximativo proposto por [Li e Zhao, 2007], quando, para cada tarefa i , os tempos de A_i , l_i e B_i são iguais; um algoritmo 3-aproximativo, também proposto por [Li e Zhao, 2007], quando apenas a duração A_i é diferente de uma tarefa para outra e um algoritmo 3-aproximativo para a versão “flow-shop”, proposto por [Leung e Zhao, 2007].

3. Metodologia

A versão original CTSP, que será tratada nesse artigo, é um problema de estrutura muito simples, em termos de quantidade de dados. Em uma instância, existem apenas N tarefas, cada uma com o tempo de duração das suas duas etapas e um tempo de intervalo. No entanto, como foi dito na Seção 2, existem casos em que podem ser aplicados algoritmos específicos que encontram uma solução garantidamente próxima à solução ótima.

Esse trabalho busca investigar se essas situações específicas podem ser percebidas durante a execução de modelagens matemáticas por parte de otimizadores bem conhecidos. Em outras palavras, o que se pretende é verificar se os otimizadores conseguem, de alguma forma, se aproveitar das características dos casos específicos para encontrar uma solução ótima de forma mais eficiente ou não.

Para tanto, serão testadas duas formulações matemáticas de Programação Inteira Mista, existentes na literatura ([Khatami et al., 2020]), e uma formulação de Programação por Restrições, proposta nesse trabalho. A Programação por Restrições ([Ochoa et al., 2019],[Wang et al., 2015]) é uma abordagem bem mais abrangente do que a Programação Linear/Inteira, pois ela permite outros tipos de variáveis como variáveis lógicas (verdadeiro/falso), variáveis semi-contínuas, dentre outras.

Na prática e de forma bem resumida, a execução de uma modelagem com Programação por Restrições começa com uma análise do domínio inicial das variáveis e das correlações que elas possuem entre si. O objetivo, nesse momento, é limitar ao máximo possível o domínio viável das variáveis, isto é, o domínio de valores que podem ocorrer em uma solução real. Quando uma variável tem o seu domínio restringido, existe a possibilidade disso impactar o domínio de outras variáveis, pois algumas combinações de valores (entre as duas) que antes eram permitidas passam a não ser mais. Quanto mais limitado é o domínio de uma variável, melhor para a execução da modelagem, pois menos combinações empregando essa variável permanecem possíveis.

Após não conseguir mais reduzir o domínio das variáveis, o otimizador passa a fazer inferências sobre um possível valor para elas, como se fosse um processo de fixação de variáveis no *branch-and-bound*. Essa “fixação” se propaga entre as variáveis correlacionadas até se obter uma solução final. Esse processo, às vezes, é chamado de busca. Se não houver alguma limitação imposta ao otimizador, ele passa a realizar buscas indefinidamente até que a solução seja a melhor possível,

3.1. Programação por Restrições

A modelagem de um problema usando Programação por Restrições não é tão padronizada quanto em Programação Linear/Inteira porque a representação das variáveis depende bastante do otimizador que será empregado. A modelagem proposta nesse trabalho foi pensada para o otimizador *CP Optimizer* da empresa IBM/Ilog. Esse otimizador permite a criação de variáveis do tipo “intervalo”, com duração pré-determinada, que são compostas por um ponto de início e um ponto de fim, ambos números reais. Esse tipo de variável é ideal para representar tarefas de um problema de escalonamento, por exemplo, como será feito com as tarefas a_i e b_i do CTSP.

$$\text{minimize} \quad \max(\text{end}(b_i)) \quad (1)$$

$$\text{s.t.} \quad \text{start}(b_i) = \text{end}(a_i) + l_i \quad i \in N, \quad (2)$$

$$\text{presence}(a_i, b_i) = \text{true} \quad i \in N, \quad (3)$$

$$\text{noOverlap}(i, i') \quad (i, i') \in N, i < i', \quad (4)$$

$$\text{length}(a_i) = A_i \quad i \in N, \quad (5)$$

$$\text{length}(b_i) = B_i \quad i \in N, \quad (6)$$

A função-objetivo (1) modela a minimização do *makespan*, que pode ser obtido pelo maior tempo de encerramento dentre todas as tarefas b_i . As restrições (2) determinam que as tarefas b_i comecem obrigatoriamente l_i unidades de tempo após o encerramento da respectiva tarefa a_i . As restrições (3) asseguram que todas as tarefas a_i e b_i serão executadas. Por fim, as restrições (4) garantem que não haverá sobreposição alguma entre as tarefas i e i' . Caso não seja possível utilizar um otimizador que possua esse tipo de restrição, será necessário estabelecer, par a par, quando uma tarefa pode começar em relação ao tempo de início da outra. As restrições (5) e (6) fixam a duração das tarefas.

3.2. Programação Inteira Mista

Também serão tratadas nesse trabalho duas formulações matemáticas já existentes na literatura, ambas na forma de Programação Inteira Mista. A primeira foi proposta originalmente por [Elshafei et al., 2004] e consiste em uma abordagem indexada por tempo. O horizonte de planejamento do problema é discretizado em um conjunto de H unidades de tempo, tal que $H = \{0..T\}$. A variável binária x_{it} vale 1 quando a tarefa a_i começa sua execução no tempo t ; caso contrário, vale 0. A constante $P_i = A_i + l_i + B_i$ denota o tempo total de processamento da tarefa i , sendo composta

do tempo necessário para a execução de suas duas partes e o tempo de atraso entre elas. Além disso, a constante $\mathcal{T}_i$ representa o maior tempo possível para que a tarefa i comece sua execução e termine dentro do horizonte de planejamento, ou seja, $\mathcal{T}_i = \mathcal{T} - P_i$. Essa formulação será chamada de F1.

A função-objetivo (7) modela a minimização do tempo de encerramento das tarefas. As restrições (8) garantem que o *makespan* seja pelo menos tão grande quanto o tempo de encerramento de cada tarefa i , considerando o início de sua execução no tempo t e seu encerramento no tempo $(t + P_i - 1)$. As restrições (9) exigem que todas as tarefas sejam executadas e encerrem dentro do horizonte de planejamento. As restrições (10) garantem que não haverá sobreposição de tarefas dentro do horizonte de planejamento. Isso ocorre porque os conjuntos $SA(i, t)$ e $SB(i, t)$ representam as unidades de tempo que serão utilizadas pelas tarefas a_i e b_i , respectivamente, quando a execução delas iniciar no tempo t .

$$\text{minimize} \quad C_{max} \quad (7)$$

$$s.t. \quad C_{max} \geq \sum_{t=0}^{\mathcal{T}_i} (t + P_i - 1) x_{it} \quad i \in N, \quad (8)$$

$$\sum_{t=0}^{\mathcal{T}_i} x_{it} = 1 \quad i \in N, \quad (9)$$

$$\sum_{i \in N} \sum_{t \in SA(i, t)} x_{it} + \sum_{i \in N} \sum_{t \in SB(i, t)} x_{it} \leq 1 \quad t \in H, \quad (10)$$

$$x_{it} \in 0, 1, \quad i \in N, \quad t \in H \quad (11)$$

A segunda formulação de Programação Inteira Mista, F2, diverge da primeira na abordagem utilizada. [Sherali e Smith, 2004] ao invés de utilizarem um modelo indexado por tempo, propuseram uma formulação que adota variáveis inteiras s_i para representar o tempo de início da tarefa i . Para evitar sobreposições, no entanto, é necessário estabelecer uma relação entre o tempo de início de duas tarefas i e i' . Existem 5 casos possíveis para essa correlação. No caso 1, a execução completa de i ocorre antes de i' começar. No caso 2, ocorre o inverso. No caso 3, durante o intervalo entre as tarefas a_i e b_i , ocorre a execução da tarefa $a_{i'}$. Esse caso só pode ocorrer quando $A_{i'} \leq l_i$ e $B_i \leq l_{i'}$. No caso 4, ocorre o inverso do caso 3, ou seja, durante o intervalo entre $a_{i'}$ e $b_{i'}$, ocorre a execução de a_i . Esse caso também só pode ocorrer quando $A_i \leq l_{i'}$ e $B_{i'} \leq l_i$. Por fim, no caso 5, a execução completa de uma tarefa ocorre durante o intervalo da outra. Para isso ocorrer, é necessário que $P_i \leq l_{i'}$ ou que $P_{i'} \leq l_i$. As duas situações não podem existir simultaneamente. As variáveis binárias $y_{ii'v}$ valem 1 se a correlação entre as tarefas i e i' for do tipo v e 0, caso contrário.

Para a formulação ficar coerente, ainda é preciso estabelecer a conexão entre as variáveis s e y . As constantes $L_{ii'v}$ e $R_{ii'v}$ representam a menor e a maior diferença de tempo possível entre o início das tarefas i e i' , para o caso v . O cálculo dessas constantes depende do caso em questão e está disponível em [Sherali e Smith, 2004].

A função-objetivo (12) modela a minimização do *makespan* do problema. As restrições (13) garantem que o *makespan* seja pelo menos tão grande quanto o tempo de início de cada tarefa i somado ao tempo total de sua execução. As restrições (14) asseguram que entre duas tarefas será aplicado apenas um modo de correlação. As restrições (15) e (16) definem que a diferença entre os tempos de início de duas tarefas i e i' será, respectivamente, maior ou igual ao mínimo possível ($L_{ii'v}$) e menor ou igual ao máximo possível ($R_{ii'v}$), para a correlação escolhida v . As restrições (17) e (17) indicam o domínio das variáveis envolvidas.

$$\text{minimize} \quad C_{max} \quad (12)$$

$$\text{s.t.} \quad C_{max} \geq s_i + P_i \quad i \in N, \quad (13)$$

$$\sum_{v=1}^5 y_{ii'v} = 1 \quad (i, i') \in N, \quad i < i', \quad (14)$$

$$s_{i'} - s_i \geq \sum_{v=1}^5 L_{ii'v} y_{ii'v} \quad (i, i') \in N, \quad i < i', \quad (15)$$

$$s_{i'} - s_i \leq \sum_{v=1}^5 R_{ii'v} y_{ii'v} \quad (i, i') \in N, \quad i < i', \quad (16)$$

$$s_i \geq 0 \quad i \in N, \quad (17)$$

$$y_{ii'v} \in \{0, 1\}, \quad (i, i') \in N, \quad i < i', \quad v \in \{1, 2, 3, 4, 5\} \quad (18)$$

4. Resultados Computacionais

Os testes computacionais foram feitos utilizando um processador i5-12450H, com 16GB de memória principal, armazenamento secundário em SSD e sistema Windows 11. As instâncias de teste foram obtidas de <https://data.mendeley.com/datasets/dd7ht5k5pn/1> e propostas por [Khatami et al., 2020]. Foram escolhidos 5 conjuntos de instâncias com características distintas com o propósito de verificar quais dessas características tornam a resolução da instância mais ou menos eficiente, em termos práticos. Cada conjunto é formado por grupos de 150 instâncias com tamanhos de 5, 10, 15, 20 ou 25 tarefas cada (30 instâncias de cada tamanho).

A formulação de Programação por Restrições utilizou o *CP Optimizer 22.1.1.0* enquanto as formulações de Programação Inteira Mista usaram o *CPLEX 22.1.1.0*. As configurações-padrão foram utilizadas, exceto pelo tempo máximo de 900 segundos por execução.

O primeiro conjunto de instâncias não apresenta nenhuma característica específica, ou seja, os três componentes das tarefas (duração da primeira parte, duração do intervalo e duração da segunda parte) não seguem nenhuma regra e podem variar de valor entre uma tarefa e outra. Para esse tipo de instância, [Shapiro, 1980] e [Sherali e Smith, 2004] mostraram que o problema é NP-Difícil.

Os resultados podem ser vistos na Tabela 1. Nas tabelas deste trabalho, a primeira coluna representa o grupo de 30 instâncias em questão. As colunas “TM(s)” representam o tempo médio (em segundos) da execução dessas instâncias. As colunas “GM(%)” indicam o *gap* médio (em percentual) entre a melhor solução encontrada e o melhor limite dual. As colunas “SO” indicam a quantidade de soluções ótimas encontradas e provadas no grupo.

Tabela 1: Instâncias sem qualquer restrições

Grupo N	TM(s)			GM(%)			SO		
	CP	F1	F2	CP	F1	F2	CP	F1	F2
5	0,1	8,5	0,1	0,0	0,0	0,0	30	30	30
10	3,1	791,3	4,8	0,0	13,9	0,0	30	7	30
15	618,7	900,0	859,1	1,0	38,1	20,6	21	0	2
20	900,0	900,0	900,0	3,5	56,4	58,0	0	0	0
25	900,0	900,0	900,0	3,7	69,1	72,6	0	0	0

Tabela 2: Instâncias onde todas as tarefas a_i têm a mesma duração

Grupo N	TM(s)			GM(%)			SO		
	CP	F1	F2	CP	F1	F2	CP	F1	F2
5	0,1	8,5	0,0	0,0	0,0	0,0	30	30	30
10	3,5	713,0	8,4	0,0	8,6	0,0	30	8	30
15	646,4	900,0	871,3	1,8	35,9	30,0	17	0	2
20	900,0	900,0	900,0	3,9	55,6	62,8	0	0	0
25	900,0	900,0	900,0	3,8	64,7	72,7	0	0	0

Tabela 3: Instâncias onde todos os intervalos l_i têm a mesma duração

Grupo N	TM(s)			GM(%)			SO		
	CP	F1	F2	CP	F1	F2	CP	F1	F2
5	0,1	48,9	0,0	0,0	0,0	0,0	30	30	30
10	28,1	709,3	11,2	0,0	16,1	0,0	30	7	30
15	573,0	900,0	893,1	2,5	41,4	25,5	18	0	1
20	900,0	900,0	900,0	3,7	57,5	59,3	0	0	0
25	900,0	900,0	900,0	3,7	66,2	75,0	0	0	0

De acordo com a Tabela 1, é possível perceber uma significativa diferença entre os *gaps* médios da formulação CP para as outras. Isso é explicado pelo limite dual que melhora muito rapidamente durante a execução. Tipicamente, o limite dual alcança seu valor final em poucos segundos, sendo o restante da execução mais focado em encontrar soluções melhores. Os limites duais das formulações F1 e F2 foram 29% piores, em média, em relação à formulação CP, chegando a aproximadamente 45% pior, apenas nas instâncias onde a solução ótima não foi provada.

Outro fator a ser considerado diz respeito à melhor solução encontrada durante a execução. A formulação CP encontrou a melhor solução em todas as 150 instâncias, enquanto F1 encontrou em 40 instâncias e F2 em 62 instâncias, considerando empates. Ou seja, a dificuldade mais comum encontrada pela formulação CP foi apenas em provar a otimalidade da solução incumbente, o que pode ser explicado pelo algoritmo do otimizador, que vai realizando inferências e buscas até esgotar todas as possibilidades disponíveis. Esse é um processo que pode ser bastante demorado, dependendo da instância. F1 e F2 tiveram soluções 24% e 4% piores em média, respectivamente, nas instâncias onde a solução ótima não foi provada.

As Tabelas 2 e 3 mostram os resultados para o conjunto de instâncias onde todas as tarefas a_i e todos os intervalos l_i , respectivamente, têm a mesma duração. Embora [Orman e Potts, 1997] tenham mostrado que instâncias desse tipo podem ser consideradas NP-Difícil, era esperado que a execução dos otimizadores sofresse alguma influência dessas características.

Os resultados das Tabelas 2 e 3, no entanto, não mostraram nenhuma diferença significativa em relação aos resultados da Tabela 1. A formulação CP continua com desempenho melhor tanto em termos de *gap* médio quanto em número de soluções ótimas provadas.

Além disso, um resultado negativo passou a acontecer com a execução da formulação F1. Nas instâncias do grupo com $N = 20$ e $N = 25$ da Tabela 2, o CPLEX não conseguiu sequer terminar de explorar o nó raiz do algoritmo do *branch-and-bound* em 2 e 9 instâncias, respectivamente. A quantidade de variáveis binárias da formulação é $n * \mathcal{T}$, isto é, depende do tamanho do horizonte de planejamento. Esse, sem dúvidas, é um ponto fraco dessa formulação, pois o cálculo ruim do tamanho do horizonte de planejamento pode fazer com que a quantidade de variáveis binárias e,

Tabela 4: Instâncias onde todas as tarefas a_i e b_i têm duração de uma unidade de tempo ($A_i = B_i = 1$)

Grupo N	TM(s)			GM(%)			SO		
	CP	F1	F2	CP	F1	F2	CP	F1	F2
5	0,0	0,1	0,0	0,0	0,0	0,0	30	30	30
10	0,0	0,2	0,0	0,0	0,0	0,0	30	30	30
15	0,0	0,4	0,2	0,0	0,0	0,0	30	30	30
20	0,0	0,8	1,7	0,0	0,0	0,0	30	30	30
25	0,0	1,5	24,8	0,0	0,0	0,0	30	30	30

consequentemente, a matriz de coeficientes que será manipulada pelo algoritmo *branch-and-bound* seja muito grande.

O cálculo utilizado no experimento foi $\mathcal{T} = \sum_{i=1}^N P_i$, onde $P_i = A_i + l_i + B_i$, ou seja, o tempo total de todas as tarefas somadas. Infelizmente, não é do nosso conhecimento uma forma mais precisa ou mais ajustada de calcular esse horizonte de planejamento, pois em algumas instâncias, os intervalos das tarefas são tão pequenos que não conseguem ser aproveitados para se começar ou terminar a execução de outras tarefas.

Com essa situação e com a matriz de coeficientes relativamente densa, o processamento do nó raiz acaba demorando muito em algumas instâncias. A possibilidade de “desligar” o processamento mais intensivo do nó raiz foi testada, mas os resultados mostraram um limite dual muito pior, sem resultados positivos em relação ao *gap* final ou à qualidade da melhor solução encontrada durante o limite de tempo.

O mesmo problema ocorreu nas instâncias onde o intervalo é mesmo para todas as tarefas, cujos resultados foram mostrados na Tabela 3. Nesse caso, nos grupos com $N = 15$, $N = 20$ e $N = 25$, foram observadas 1, 8 e 8 instâncias, respectivamente, nas quais a execução da formulação F2 encerrou antes do término da exploração do nó raiz.

Vale destacar que, para as formulações CP e F2, não é preciso definir o tamanho máximo do horizonte de planejamento, pois a representação do início (e do fim) da execução das tarefas é feita por variáveis inteiras que são independentes de índices temporais.

O próximo conjunto de instâncias testadas apresenta uma característica bastante peculiar: a duração das tarefas a_i e b_i é sempre igual a uma unidade de tempo ($A_i = B_i = 1$). Essa característica é costumeiramente chamada de UET (*Unity Execution Time*). [Békési et al., 2009] mostraram que existe um algoritmo p -aproximativo, onde $\frac{28}{19} \leq p \leq \frac{7}{4}$. A classe dessas instâncias continua sendo NP-difícil. Os resultados podem ser vistos na Tabela 4.

Os resultados da Tabela 4 mostram que esse tipo de instância é resolvido muito facilmente pelos otimizadores utilizados. Como o tempo de execução das tarefas a_i e b_i é mínimo, basta encontrar uma sequência de execução que não cause sobreposições, o que torna o escalonamento bastante flexível. O tempo um pouco maior gasto pela formulação F2 para as instâncias com $N = 25$ se deve ao processo de fixação das variáveis inteiras, que pode não ser tão imediato quanto nas variáveis binárias. Mesmo assim, os tempos são considerados bem baixos, caso a solução ótima seja imprescindível.

Por fim, o último conjunto testado possui instâncias compostas por N tarefas idênticas, ou seja, as durações de a_i , l_i e b_i são as mesmas para todas as tarefas da instância em questão. Para esse caso específico, existe um algoritmo $\mathcal{O}(\log n)$, proposto por [Baptiste, 2010]. Desta maneira, a utilização de formulações matemáticas não é a abordagem mais eficiente em termos práticos, pois todo o processo de criação e execução do modelo poderia ser substituído pela execução direta do algoritmo. Em instâncias de maior porte, o tempo consumido pelas etapas de preparação, leitura e

Tabela 5: Instâncias onde todas as tarefas têm a mesma duração e o mesmo intervalo (sem quebra de simetria)

Grupo N	TM(s)			GM(%)			SO		
	CP	F1	F2	CP	F1	F2	CP	F1	F2
5	0,2	9,7	0,1	0,0	0,0	0,0	30	30	30
10	151,7	86,5	360,0	1,9	2,9	4,8	28	28	21
15	873,1	418,9	540,1	15,9	12,3	32,4	1	18	12
20	900,0	610,3	510,1	15,6	27,7	40,1	0	11	13
25	900,0	571,2	482,5	16,3	33,9	41,8	0	13	14

pré-processamento da modelagem começaria a ser muito maior.

O que se pretende avaliar aqui é se os otimizadores conseguem encontrar um mecanismo de resolver mais eficientemente a modelagem matemática, lançando mão da similaridade das tarefas. Os resultados são apresentados na Tabela 5.

De forma contra-intuitiva, até certo ponto, a formulação CP que vinha obtendo bons resultados nos testes anteriores teve o pior desempenho dentre as três modelagens testadas. O desempenho ruim se deve aos limites duais que não foram aprimorados tão bem quanto antes. Esse foi o primeiro conjunto de instâncias onde F1 e F2 tiveram limites duais melhores, em algumas instâncias. Por outro lado, as formulações CP e F2 obtiveram a melhor solução factível em 150 e 142 instâncias, respectivamente (considerando empates), enquanto a melhor solução foi obtida por F1 em apenas 116 casos.

A execução em si da formulação CP também apresentou um comportamento diferente, em relação aos testes anteriores. A melhor solução final foi encontrada em menos de 1 segundo, em 148 instâncias. A partir desse ponto, todas as inferências e buscas realizadas pelo otimizador surtiram pouco efeito nas instâncias maiores, pois o otimizador “estagnou”, sem conseguir melhorar nem a solução incumbente nem o limite dual.

As soluções geradas após a “estagnação”, embora distintas em termos de valores de variáveis, apresentam o mesmo resultado estrutural final, pois as tarefas são idênticas. Assim, executar as tarefas do problema em qualquer ordem vai gerar a mesma solução, apenas com as variáveis trocadas. Essa situação é comumente chamada de problema de simetria e é bastante comum em problemas envolvendo grafos ([Talbi e Weinberg, 2007]) ou clusterização ([Pelikan e Goldberg, 2000]).

Nesses casos, uma forma de auxiliar a execução da formulação consiste em acrescentar ao modelo restrições específicas que quebrem a simetria, sem comprometer a busca por soluções factíveis. A quebra de simetria proposta nesse trabalho consiste apenas em definir uma ordenação completa para a execução das tarefas, sem forçar que seu início ocorra em um determinado tempo (a definição do tempo de início das tarefas é um trabalho do otimizador). Foi estabelecido que a tarefa i deve iniciar antes da tarefa $i + 1$, para todas as tarefas de cada instância. Assim, a otimização não explorará soluções diferentes que tenham apenas a ordem das tarefas trocada, ou seja, soluções que sejam apenas uma permutação da ordem de execução das tarefas.

4.1. Quebra de Simetria

As restrições (19), (20) e (21) foram adicionadas às formulações CP, F1 e F2, respectivamente, para garantir que a execução de uma tarefa i ocorra antes da tarefa $i + 1$. As formulações expandidas, com quebra de simetria, foram executadas novamente para as mesmas instâncias da Tabela 5 e os novos resultados podem ser vistos na Tabela 6.

Tabela 6: Instâncias onde todas as tarefas têm a mesma duração e o mesmo intervalo (com quebra de simetria)

Grupo N	TM(s)			GM(%)			SO		
	CP	F1	F2	CP	F1	F2	CP	F1	F2
5	0,0	0,8	0,0	0,0	0,0	0,0	30	30	30
10	0,1	72,2	0,0	0,0	1,2	0,0	30	28	30
15	0,1	472,4	0,1	0,0	-	0,0	30	17	30
20	0,1	778,6	0,2	0,0	-	0,0	30	6	30
25	0,3	790,8	2,9	0,0	-	0,0	30	4	30

$$start(a_i) < start(a_{i+1}) \quad i \in \{1, \dots, |N| - 1\}, \quad (19)$$

$$\sum_{t=0}^{\mathcal{T}_i} t * x_{it} < \sum_{t=0}^{\mathcal{T}_i} t * x_{i+1t} \quad i \in \{1, \dots, |N| - 1\}, \quad (20)$$

$$s_i < s_{i+1} \quad i \in \{1, \dots, |N| - 1\}, \quad (21)$$

Os resultados obtidos pelos otimizadores para as formulações com quebra de simetria mostram que as novas restrições foram muito efetivas para as formulações CP e F2, que passaram a encontrar e provar a solução, para todas as 150 instâncias testadas, em poucos segundos. De fato, o desempenho original ruim, sobretudo da formulação CP, nesse conjunto de instâncias com tarefas idênticas se devia à simetria que foi resolvida pela adição das novas restrições.

Contudo, a formulação F1 mostrou um desempenho ainda pior com a adição das restrições. Em 47 das 150 instâncias, a execução da formulação não conseguiu gerar sequer uma solução factível dentro do tempo limite, o que inviabilizou o cálculo do *gap* médio e dificultou a obtenção da solução ótima em algumas instâncias.

5. Conclusões

O presente trabalho abordou o Coupled-Task Scheduling Problem (CTSP). O objetivo do trabalho foi investigar se características específicas das tarefas podem fazer com que a execução de modelagens matemáticas de Programação Inteira Mista (F1 e F2) e Programação por Restrições (CP) apresente resultados diferentes significativos.

Os resultados mostraram que algumas características tiveram impacto, sim, como as tarefas contendo apenas uma unidade de tempo de execução. No caso de tarefas idênticas, a execução das formulações originais foi afetada pelo problema de simetria. A adição de restrições que quebram a simetria foi bem aproveitada pelas formulações CP e F2, mas não pela formulação F1, que teve sua execução prejudicada com as novas restrições impostas.

De modo geral, a formulação CP, proposta nesse trabalho, obteve resultados melhores que as demais formulações da literatura em todos os conjuntos de instâncias testadas e também apresentou bom desempenho quando foi alterada para evitar o problema de simetria. Como pontos positivos da formulação, é válido destacar ainda a grande capacidade de encontrar as melhores soluções (comparada às outras duas formulações) e o baixo *gap* médio obtido. O tempo médio de execução da formulação F2 é outro ponto importante a ser destacado.

6. Trabalhos Futuros

Como trabalhos futuros, ainda são possíveis testes com instâncias que possuam outras características além daquelas que foram testadas, bem como instâncias de maior porte. O acréscimo de restrições que fortaleçam as formulações e façam o limite dual ser mais aprimorado ao longo da execução também é uma boa oportunidade de estudo.

Referências

- Ageev, A. e Kononov, A. (2007). Approximation algorithms for scheduling problems with exact delays. In *Proceedings of the approximation and online algorithms*, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Allahverdi, A. (2016). A survey of scheduling problems with no-wait in process. *European Journal of Operational Research*, 255(3):665–686.
- Arabameri, S. e Salmasi, N. (2004). Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem. *Computers & Industrial Engineering*, 64(4):902–916.
- Baptiste, P. (2010). A note on scheduling identical coupled tasks in logarithmic time. *Discrete Applied Mathematics*, 158(5):583–587. 10.1016/j.dam.2009.10.012.
- Blazewicz, J., Ecker, K., Kis, T., Potts, C., Tanas, M., e Whitehead, J. (2010). Scheduling of coupled tasks with time unit processing times. *Journal of Scheduling*, 13(5):453–461. 10.1007/s10951-010-0167-z.
- Blazewicz, J., Pawlak, G., Tanas, M., e Wojciechowski, W. (2012). New algorithms for coupled tasks scheduling - a survey. *RAIRO-Operations Research*, 46(4):335–353. 10.1051/ro/20122020.
- Brauner, N., Finke, G., Lehoux-Lebacque, V., Potts, C., e Whitehead, J. (2009). Scheduling of coupled tasks and one-machine no-wait robotic cells. *Computers & Operations Research*, 36(2): 301–307. 10.1016/j.disopt.2005.08.002.
- Békési, J., Galambos, G., Jung, M., Oswald, M., e Reinelt, G. (2014). A branch-and-bound algorithm for the coupled task problem. *Mathematical Methods of Operations Research*, 80(1): 47–81. 10.1007/s00186-014-0469-6.
- Békési, J., Galambos, G., Oswald, M., e Reinelt, G. (2009). Improved analysis of an algorithm for the coupled task problem with uet jobs. *Operations Research Letters*, 37(2):93–96. 10.1016/j.orl.2008.11.002.
- Condotta, A. e Shakhlevich, N. (2012). Scheduling coupled-operation jobs with exact time-lags. *Discrete Applied Mathematics*, 160(16):2370–2388–361. 10.1016/j.dam.2012.05.026.
- Condotta, A. e Shakhlevich, N. (2014). Scheduling patient appointments via multilevel template: A case study in chemotherapy. *Operations Research for Health Care*, 3(3):129–144. 10.1016/j.orhc.2014.02.002.
- Dell’Amico, M. (1996). Shop problems with two machines and time-lags. *Operations Research*, 44(5):777–787.
- Elshafei, M., Sherali, H., e Smith, J. (2004). Radar pulse interleaving for multi target tracking. *Naval Research Logistics (NRL)*, 51(1):72–94. 10.1002/nav.10103.

- Hwang, F. e Lin, B. (2004). Coupled-task scheduling on a single machine subject to a fixed-job-sequence. *Computers & Industrial Engineering*, 60(4):690–698. 10.1016/j.cie.2011.01.002.
- Khatami, M., Salehipour, A., e Cheng, T. (2020). Coupled task scheduling with exact delays: Literature overviews and models. *European Journal of Operational Research*, 282(1):19–39. 10.1016/j.ejor.2019.08.045.
- Leung, J.-T. e Zhao, H. (2007). Scheduling two-machine flow shops with exact delays. *International Journal of Foundations of Computer Science*, 18(2):341–359. 10.1142/S0129054107004711.
- Li, H. e Zhao, H. (2007). Scheduling coupled-tasks on a single machine. In *Proceedings of the IEEE symposium on computational intelligence in scheduling*, p. 137–142. 10.1109/SCIS.2007.367681.
- Ochoa, L., González-Rojas, O., Cardozo, N., González, A., Chavarriaga, J., Casallas, R., e Díaz, J. (2019). Constraint programming heuristics for configuring optimal products in multi product lines. *Information Sciences*, 474:33–47. 10.1016/j.ins.2018.09.042.
- Orman, A. e Potts, C. (1997). On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, 72(1):141–154.
- Orman, A., Potts, C., Shahani, A., e Moore, A. (1996). Scheduling for a multifunction phased array radar system. *European Journal of Operational Research*, 90(1):13–25.
- Pelikan, M. e Goldberg, D. E. (2000). Genetic algorithms, clustering, and the breaking of symmetry. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., e Schwefel, H.-P., editors, *Parallel Problem Solving from Nature PPSN VI*, p. 385–394, Berlin, Heidelberg. Springer Berlin Heidelberg. ISBN 978-3-540-45356-7. 10.1007/3-540-45356-3-38.
- Potts, C. e Whitehead, J. (2007). Heuristics for a coupled-operation scheduling problem. *Journal of the Operational Research Society*, 58(10):1357–1388. 10.1057/palgrave.jors.2602272.
- Shapiro, R. (1980). Scheduling coupled tasks. *Naval Research Logistics Quarterly*, 27(3):489–498. 10.1002/nav.3800270312.
- Sherali, H. e Smith, J. (2004). Interleaving two-phased jobs on a single machine. *Discrete Optimization*, 2(4):248–361. 10.1016/j.disopt.2005.08.002.
- Simonin, G., Giroudeau, R., e König, J.-C. (2011). Complexity and approximation for scheduling problem for a torpedo. *Computers & Industrial Engineering*, 61(2):352–356.
- Simonin, G., Giroudeau, R., e König, J.-C. (2013). Approximating a coupled-task scheduling problem in the presence of compatibility graph and additional tasks. *International Journal of Planning and Scheduling*, 1(4):285–300. 10.1504/IJPS.2013.059680.
- Talbi, E.-G. e Weinberg, B. (2007). Breaking the search space symmetry in partitioning problems: An application to the graph coloring problem. *Theoretical Computer Science*, 378(1):78–86. 10.1016/j.tcs.2007.01.023.
- Wang, T., Meskens, N., e Duvivier, D. (2015). Scheduling operating theatres: Mixed integer programming vs. constraint programming. *European Journal of Operational Research*, 247(2):401–413. 10.1016/j.ejor.2015.06.008.