UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE
DEPARTAMENTO DE COMPUTAÇÃO

JOÃO ANTONIO DE SOUZA

CLUSTERING SEARCH APLICADO AO PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS

ALEGRE - ES
2023

JOÃO ANTONIO DE SOUZA

CLUSTERING SEARCH APLICADO AO PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Dayan de Castro Bissoli.

ALEGRE - ES

2023

JOÃO ANTONIO DE SOUZA

CLUSTERING SEARCH APLICADO AO PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 21 de julho de 2023.

COMISSÃO EXAMINADORA

Prof. Dr. Dayan de Castro Bissoli
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Geraldo Regis Mauri
Universidade Federal do Espírito Santo

Prof. Dr. Edmar Hell Kampke
Universidade Federal do Espírito Santo

AGRADECIMENTOS

Gostaria de expressar minha gratidão a todas as pessoas que contribuíram e tornaram possível a conclusão deste trabalho. Seus apoios, incentivos e suporte foram fundamentais para alcançar este objetivo.

Primeiramente, quero agradecer à minha família, que sempre esteve ao meu lado, me apoiando incondicionalmente em todas as etapas da minha jornada acadêmica. Obrigado por serem minha fonte de força e motivação, por acreditarem em mim quando eu duvidava de mim mesmo e por me incentivarem a nunca desistir.

Agradeço aos meus amigos do curso, que compartilharam comigo os desafios e as alegrias ao longo desses anos. Nossas discussões, trocas de ideias e colaboração mútua foram fundamentais para o meu crescimento acadêmico e pessoal.

Em especial, gostaria de mencionar o meu amigo Harrison. Seu apoio constante, paciência e habilidade em explicar conceitos complexos foram essenciais para o desenvolvimento deste trabalho. Suas sugestões e insights valiosos foram fundamentais para a qualidade final deste TCC.

Também quero expressar minha gratidão ao meu orientador, Dayan, por sua orientação excepcional ao longo de todo o processo de pesquisa e redação deste trabalho. Seu comprometimento em me guiar na direção certa e suas contribuições críticas foram inestimáveis para o sucesso deste estudo.

Agradeço a todos os professores e colaboradores do curso, que compartilharam seus conhecimentos e experiências, enriquecendo meu aprendizado durante toda a graduação.

Por fim, agradeço a todos que, de alguma forma, contribuíram para a realização deste trabalho, mesmo que não estejam mencionados nominalmente. Cada incentivo, conversa ou conselho foi importante para o meu crescimento acadêmico e pessoal.

Obrigado a todos vocês por fazerem parte desta jornada e por me ajudarem a alcançar este marco em minha vida acadêmica.

RESUMO

O problema de escalonamento de projetos com restrição de recursos (da sigla em inglês RCPSP) vem sendo estudado a mais de duas décadas, e ainda tem desafiado muitos pesquisadores no mundo todo. Ele consiste em encontrar um cronograma de atividades viável, utilizando um número limitado de recursos, tal que sua duração total seja a menor possível. Apesar de existirem método exatos para sua solução, eles ainda não são capazes de resolver problemas de grande escala de maneira satisfatória. Logo, as abordagens heurísticas continuam sendo a única maneira viável para solucioná-los. Este estudo se propõe a aplicar e avaliar o desempenho da meta-heurística híbrida *Clustering Search* (CS) para resolver o RCPSP clássico, uma vez que o método ainda não foi utilizado para este fim, e, apresentou bons resultados para vários problemas de otimização NP-difícil. Os resultados obtidos mostram que o CS apresentou um bom resultado para as instâncias menores, com um potencial encorajador para problemas de maior escala.

Palavras-chave: Problema de escalonamento de projetos com restrição de recursos; RCPSP; Meta-heurística híbrida; *Clustering Search.*

ABSTRACT

The Resource-Constrained Project Scheduling Problem (RCPSP) has been studied for over two decades and still poses a challenge for many researchers worldwide. It consists of finding a feasible schedule of activities, using a limited number of resources, in such a way that its total duration is minimized. Although there are exact methods for its solution, they are still not capable of satisfactorily solving large-scale problems, thus heuristic approaches remain the only viable way to solve them. This study aims to apply and evaluate the performance of the hybrid metaheuristic called Clustering Search (CS) to solve the classic RCPSP, as this method has not been used for this purpose before but has shown promising results for various NP-hard optimization problems. The obtained results show that CS performed well for smaller instances, with encouraging potential for larger-scale problems.

Keywords: Resource-Constrained Project Scheduling Problem; RCPSP; Hybrid Metaheuristic; Clustering Search.

LISTA DE FIGURAS

Figura 1 - Exemplo de projeto..... 8
Figura 2 - Cronograma viável..... 8
Figura 3 – Componentes do CS.....14
Figura 4 - Peseudo código do algorítmo proposto.....17
Figura 5 – Exemplo de inserção de atividades.....18
Figura 6 – Exemplo de correção da posição da atividade.....19
Figura 7 – Distância de Hamming entre duas soluções..... 20

LISTA DE TABELAS

Tabela 1 - Valores dos parâmetros do CS..... 23
Tabela 2 - Resultados obtidos pelo CS para as instâncias J30, J60 e J120 selecionadas.... 23
Tabela 3 - Comparação dos resultados obtidos pelo CS com os melhores resultados da literatura..... 25

SUMÁRIO

1. INTRODUÇÃO 6
1.1. O PROBLEMA E SUA IMPORTÂNCIA 7
1.2. OBJETIVOS 9
1.2.1. Objetivo geral 9
1.2.2. Objetivos específicos 9
1.3. ESTRUTURA DO TRABALHO 9
2. REVISÃO DA LITERATURA 10
2.1. TRABALHOS CORRELATOS AO RCPSP 11
2.2. ESTADO DA ARTE DO RCPSP 12
3. METODOLOGIA 13
3.1. O CLUSTERING SEARCH 13
4. O ALGORITMO CS PROPOSTO 15
4.1. A CONSTRUÇÃO DE SOLUÇÕES 18
4.2. A SOLUÇÃO INICIAL 19
4.3. GERAÇÃO DE SOLUÇÕES VIZINHAS 19
4.4. O CÁLCULO DE SIMILARIDADE 20
4.5. BUSCA LOCAL 20
5. TESTES COMPUTACIONAIS 22
5.1. A DEFINIÇÃO DOS PARÂMETROS 22
6. RESULTADOS 23
6.1. RESULTADOS COMPUTACIONAIS 23
6.2. COMPARAÇÃO COM A LITERATURA 25
7. CONCLUSÃO 26
REFERÊNCIAS 27

1. INTRODUÇÃO

Introduzido por Pritsker *et al*. (1969), o Problema de Escalonamento de Projetos Com Restrição de Recursos (R*esource-Constrained Project Scheduling Problem - RCPSP)* ocupa um lugar central na área de gerenciamento de projetos, e deu origem a um extenso corpo de pesquisa (PELLERIN, *et al*, 2020). Este problema busca encontrar um cronograma viável, minimizando a duração total (*makespan*) de um projeto que consiste em um conjunto de atividades, com durações conhecidas, podendo as atividades estarem sujeitas a restrições de precedência e disponibilidade de recursos (CHEN *et al*., 2010).

Muitas pesquisas foram desenvolvidas na tentativa de resolver o RCPSP de forma exata. No entanto, pelo problema ser considerado NP-difícil, esses estudos só conseguem resolver, de maneira satisfatória, problemas de pequena escala com até 60 atividades. Logo, as soluções propostas por procedimentos heurísticos continuam sendo a única maneira viável de solucionar problemas com proporções maiores (CHEN *et al*., 2010; PELLERIN *et al*. 2020).

Uma extensa revisão bibliográfica foi realizada por Pellerin *et al* (2020), sobre a utilização de meta-heurísticas híbridas para solucionar o RCPSP, segundo eles, o interesse crescente em pesquisa operacional por meio de meta-heurísticas, resultou em uma tendência de se resolver o RCPSP utilizando métodos heurísticos híbridos (que fazem união de meta-heurísticas), ao invés de meta-heurísticas clássicas. Neste trabalho, podemos observar que os resultados obtidos para as instâncias J30, J60 e J120 da conhecida biblioteca PSPLIB, foram melhores nos trabalhos de Chen *et al*. (2010), Proon e Jin (2011), Paraskevopoulos, Tarantilis e Ioannou (2012) e Elsayed *et al*. (2017).

A proposta deste trabalho é avaliar os resultados obtidos pela meta-heurística híbrida *Clustering Search* (CS) quando aplicada ao RCPSP, que até o momento não foi utilizada para este fim. A justificativa para a escolha se dá pelo bom desempenho do CS em problemas NP-difícil como mostrado por Oliveira, Chaves e Lorena (2013), que realizou um levantamento de estudos utilizando o CS aplicado a uma grande variedade de problemas, onde o método apresentou resultados competitivos e muitas vezes melhores do que os encontrados na literatura. Podemos observar também em estudos mais recentes que o CS continua a apresentar bons resultados

como os apresentados em: Alvarenga Rosa *et al.* (2016) para o *Capacitated Helicopter Routing Problem*, Abreu *et al.* (2020) para o *Network Sensor Location Problem* e em Bissoli, Zufferey e Amaral (2021) para o *Flexible Job Shop Scheduling Problem*.

1.1. O PROBLEMA E SUA IMPORTÂNCIA

O objetivo do RCPSP é encontrar o cronograma de um conjunto de atividades tal que sua duração total (*makespan*), seja a menor possível enquanto satisfaz uma série de restrições. Um projeto consiste em um conjunto de atividades V = {0, 1, …, *n*, *n+1*} e K tipos de recursos renováveis. A duração de uma atividade j em V é dada como d_j e a disponibilidade de cada tipo de recurso k ∈ K em cada passo de tempo é dado como R_k e cada atividade *j* requer r_{jk} unidades do recurso k durante cada período de sua duração. Duas atividades fictícias (0 e n+1) são utilizadas para representar o início e o fim do projeto, onde $d_0 = d_{n+1} = 0$. (CHEN *et al*. 2010; ELSAYED *et al*., 2017).

As seguintes restrições serão consideradas neste trabalho:

1. Restrição de precedência: uma atividade j não pode ser iniciada até que todas as suas antecessoras sejam concluídas, logo, o tempo de iniciação desta atividade sempre será maior ou igual ao tempo de finalização de todas as suas antecessoras.
2. Restrição de recurso: a soma das necessidades de recursos para o recurso k não pode exceder R_k.

Sendo o tempo de conclusão de j denotado por F_j, os tempos de conclusão de um cronograma S são denotados como F_1, F_2, … F_n. O modelo conceitual do RCPSP pode ser descrito da seguinte forma (CHEN *et al*. 2010):

Min F_{n+1} (1)

Sujeito a:

$$F_j \leq F_{j+1} - d_j, \forall\, j = \{1, 2, ..., n+1\} \quad (2)$$

$$\sum_{j \in A(t)} r_{kj} \leq R_k, \ \forall\, k = \{1, 2, ..., K\} \quad (3)$$

$$F_j \geq 0, \forall\, k = \{1,2, ..., K\} \quad (4)$$

A Eq. (1) diz respeito à função objetivo que deve ser minimizada. Na Eq. (2) são impostas as restrições de precedência entre as atividades, enquanto na Eq. 3 são impostas a restrição de limitação de recursos onde $A(t) = j \in V \mid F_j - d_j \leq t < F_j$. Eq. (4) descreve a restrição das variáveis de decisão (CHEN *et al*. 2010).

A Figura 1 apresenta um exemplo de projeto com 6 atividades que deve utilizar de 6 unidades de um único tipo de recurso renovável K = 1, e em seguida a Figura 2 representa o cronograma viável com um *makespan* ótimo de 15 unidades de tempo.

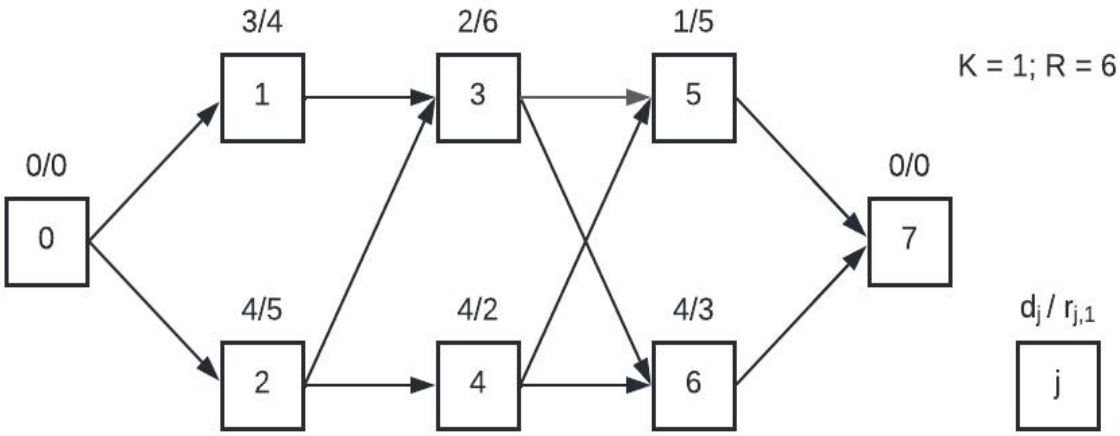

Figura 1 - Exemplo de projeto
Fonte: CHEN *et al*. (2010, p.2)

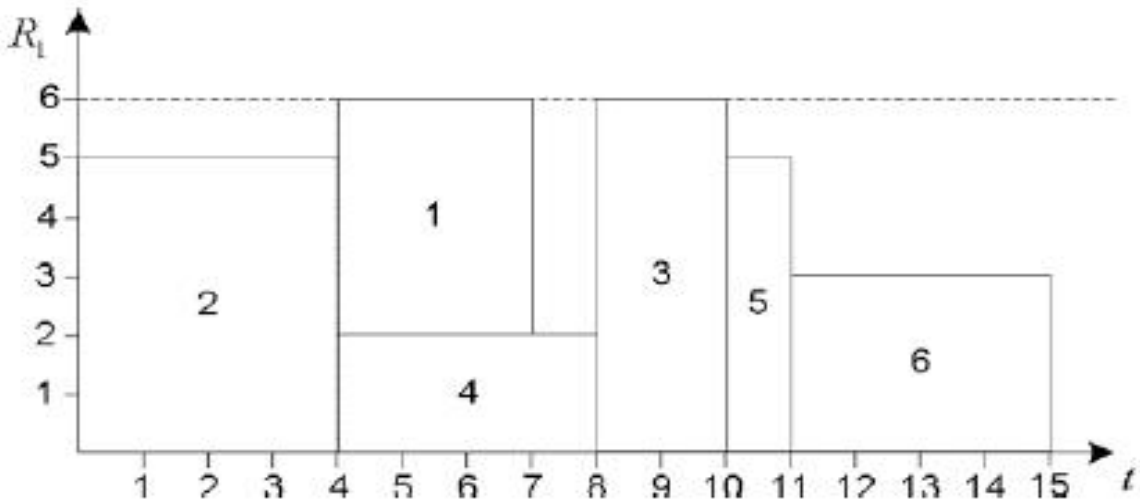

Figura 2 – Cronograma viável
Fonte: CHEN *et al*. (2010, p. 2)

1.2. OBJETIVOS

A seguir serão apresentados os objetivos gerais e específicos deste trabalho.

1.2.1. Objetivo geral

O objetivo geral é aplicar e avaliar o desempenho da meta-heurística híbrida *Clustering Search* (CS) para a resolução do problema de escalonamento de projetos com restrição de recursos (RCPSP).

1.2.2. Objetivos específicos

a) Implementar o *Clustering Search* para resolver o RCPSP.
b) Realizar experimentos computacionais utilizando o conjunto de instâncias específicas do problema clássico do RCPSP J30, J60 e J120 da biblioteca PSPLIB.
c) Comparar o desempenho do CS com os resultados apresentados pela literatura.

1.3. ESTRUTURA DO TRABALHO

A seguir, é apresentado a revisão da literatura no Capítulo 2, contendo uma breve descrição de cada trabalho relacionado a este estudo, apresentando o que foi utilizado para a solução do RCPSP e quais foram os seus resultados. A metodologia que será utilizada é descrita no Capítulo 3, onde é apresentada também a definição da meta-heurística *Clustering Search.* No Capítulo 4 é apresentado o algoritmo proposto. Os formato dos testes computacionais é apresentado no Capítulo 5. Os resultados obtidos serão apresentados no Capítulo 6, seguido pela conclusão no Capítulo 7 e as referências bibliográficas no Capítulo 8.

2. REVISÃO DA LITERATURA

As buscas pelos trabalhos correlatos ao RCPSP foram realizadas utilizando a plataforma Periódicos CAPES utilizando a palavra-chave "*resource-constrained project scheduling problem"*. Como a busca foi realizada através da ferramenta de busca avançada as palavras-chave "*metaheuristics*" e "*RCPSP*" foram adicionadas para refinar os resultados, em busca de focar mais em trabalhos que tratam de meta-heurísticas aplicadas ao problema clássico de escalonamento de projetos.

Dentre os resultados obtidos, foi encontrado o recente estudo de Pellerin, Perrier e Berthaut (2020) que consiste em uma extensa revisão bibliográfica das últimas duas décadas sobre o problema RCPSP clássico, focado na utilização de meta-heurísticas híbridas aplicadas na resolução do problema, descrevendo as principais características de cada tipo de método e em seguida comparando os resultados obtidos por cada trabalho quando aplicados às instâncias J30, J60 e J120 do problema clássico da conhecida biblioteca PSPLIB.

Segundo Pellerin, Perrier e Berthaut (2020), as medidas de desempenho seguem o protocolo experimental proposto por Kolisch (2000) e Kolisch e Hartmann (2006), onde são apresentadas as medidas de desempenho seguindo três critérios de parada: 1000, 5000 e 50000 cronogramas gerados. Os trabalhos foram classificados de acordo com o desvio médio em relação à solução ótima, ou do limitante inferior (quando o ótimo não é conhecido).

A partir desse trabalho, foram selecionados por referência cruzada os artigos que foram classificados por Pellerin, Perrier e Berthaut (2020) com os melhores resultados para serem utilizados como base de comparação neste trabalho. Por padrão, os melhores trabalhos foram aqueles que apresentaram os melhores resultados para 50000 agendamentos gerados. Os resultados para agendamentos de 5000 e 1000 foram utilizados como critério de desempate.

Os trabalhos selecionados foram: *An efficient hybrid algorithm for resource-constrained project scheduling* de Chen *et al*. (2010), *A Genetic Algorithm with Neighborhood Search for the Resource-Constrained Project Scheduling Problem* de Proon e Jin (2011), *Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm* de Paraskevopoulos, Tarantilis e Ioannou

(2012), e *Consolidated optimization algorithm for resource-constrained project scheduling problems* de Elsayed *et al*. (2017).

2.1. TRABALHOS CORRELATOS AO RCPSP

Chen *et al*. (2010) propôs um algoritmo híbrido nomeado como ACOSS, que consiste na combinação do algoritmo *Ant Colony Optimization* (ACO), um algoritmo de busca dispersa (*Scatter Search* - SS), e um algoritmo de busca local, para solucionar o RCPSP. Nesse algoritmo, é utilizado o ACO como base, e o SS é utilizado para melhorar as soluções geradas pelo ACO e atualizar as trilhas de feromônio com a melhor solução encontrada. Além disso, uma estratégia de busca local é empregada para melhorar a qualidade das soluções geradas pelo ACO, também como método de melhoria no SS. Os resultados obtidos mostram que para um número limitado de cronogramas gerados, o ACOSS foi capaz de ficar próximo das soluções ótimas, e superou outros algoritmos heurísticos para problemas de maior escala.

Proon e Jin (2011) utilizaram um algoritmo nomeado como GANS, que consiste em um algoritmo genético (GA) combinado a uma busca de vizinhança (N*eighborhood Search - NS*) para resolver o RCPSP. Nesta abordagem, o algoritmo de busca foi combinado com o GA básico modificando os operadores de cruzamento e mutação, fornecendo uma melhor direção da busca enquanto mantém a aleatoriedade da abordagem do GA. Os experimentos demonstram que a estratégia utilizada levou a um melhor desempenho do GA para resolver o RCPSP. Para instâncias de grande porte, os resultados obtidos pelo GANS foram melhores do que os de Chen *et al*. (2010), sendo inferior apenas nas instâncias J30 do PSPLIB.

Paraskevopoulos, Tarantilis e Ioannou (2012) propuseram uma nova forma de representar uma solução do RCPSP chamada de representação de lista de eventos (E*vent-List Representation*), nela, não é possível que duas representações diferentes resultem em cronogramas equivalentes (com mesmo *makespan*), fazendo com que o espaço de soluções não possua tais cronogramas, aumentando assim a eficiência da busca. O algoritmo de solução proposto é baseado no método *scatter search* que utiliza da solução de representação de lista de eventos proposta, e integra com um Busca Local Iterada Adaptativa (*Adaptive Iterated Local Search* - AILS) para melhorar a solução encontrada. Comparando os resultados obtidos,

podemos observar que o algoritmo proposto obteve bons resultados para as instâncias J30 e J60, porém sendo superior ao método GANS de Proon e Jin (2011) apenas nas instâncias J30, na J60 com 1000 agendamentos e na J120 também com 1000 agendamentos.

Elsayed *et al*. (2017) propuseram um *framework* de otimização consolidado que possui mais de um algoritmo de otimização para resolver com eficiência o RCPSP. O objetivo do método proposto é conseguir encontrar bons resultados com uma rápida taxa de convergência, uma vez que os métodos propostos na literatura, apesar de obterem bons resultados, eles nunca estão próximos do ótimo, e muitas vezes possuem uma lenta taxa de convergência. Para isso, o método utiliza de dois algoritmos multi operadores (MOA's), onde a probabilidade de aplicação de cada MOA é baseada na sua eficácia, ou seja, se um dado método está obtendo melhores resultados, uma ênfase é dada a esse método. Os resultados obtidos foram os melhores apresentados para as instâncias J30, já nas instâncias J60, o método proposto obteve melhor resultado para 1000 agendamentos, mas fica abaixo do método GANS de Proon e Jin (2011), para 5000 e 50000 agendamentos. Nas instâncias J120, todos os resultados ficaram atrás do GANS de Proon e Jin (2011), e do ACOSS de Chen *et al*. (2010).

2.2. ESTADO DA ARTE DO RCPSP

Até o momento não existe uma única pesquisa que apresentou os melhores resultados para todas as instâncias propostas pelo PSPLIB. Elsayed *et al*. (2017) foi o que apresentou os melhores resultados para as instâncias J30. O método GANS de Proon e Jin (2011), é o que obteve os melhores resultados para as instância J60 e J120, ficando atrás apenas de Paraskevopoulos,Tarantilis e Ioannou (2012) nas instâncias J60 com até 1000 cronogramas gerados, e de Chen *et al*. (2010) nas instâncias J120, também para 1000 cronogramas gerados.

3. METODOLOGIA

A metodologia deste trabalho consiste em aplicar a meta-heurística *Clustering Search* (CS), para solucionar o problema de escalonamento de projetos com restrição de recursos. O CS é uma metodologia híbrida que tem como ideia principal, identificar áreas promissoras dentro do espaço de busca utilizando uma meta-heurística, e então, agrupá-las em *clusters* que posteriormente serão explorados por uma heurística de busca local. Um *cluster* é definido por um centro c, geralmente iniciado de forma aleatória, tendendo a se posicionar em pontos promissores dentro do espaço de busca ao longo do processo. A sua cobertura é definida por uma métrica que avalia a similaridade da solução com o centro do *cluster*, ela pode ser definida pela distância euclidiana ou pelo número de movimentos necessários para se chegar ao centro do *cluster*, dependendo da natureza do problema (OLIVEIRA; CHAVES; LORENA, 2013).

3.1. O CLUSTERING SEARCH

Segundo Oliveira, Chaves e Lorena (2013), o CS pode ser dividido em quatro partes independentes:

a) A meta-heurística de busca (*Search Metaheuristic* (SM))
b) O componente de agrupamento iterativo (*Interative Clustering Component IC*)
c) O módulo analisador (*The Analyzer Module* (AM))
d) A busca local (*Local Search Heuristic* (LS))

O componente SM é utilizado para explorar o espaço de busca, nele pode ser implementada qualquer meta-heurística que consiga gerar soluções diversificadas dentro desse espaço, e é executado a cada iteração do algoritmo (OLIVEIRA; CHAVES; LORENA, 2013).

O componente IC tem como objetivo reunir soluções semelhantes em grupos, criando para eles um centro para representar o *cluster*, onde uma métrica de distância, Δ, é definida a priori, permitindo assim aferir a similaridade entre as soluções (OLIVEIRA; CHAVES; LORENA, 2013).

No componente AM, é analisado cada *cluster* em intervalos regulares a fim de identificar *clusters* promissores. Quando a densidade do *cluster*, também chamado de volume, atinge um limiar previamente definido, ele é selecionado para ser

explorado pela heurística de busca local. Uma vez que a busca local não consegue melhorar a solução do centro do *cluster*, o índice ineficácia r_j do *cluster* é incrementado, e, caso esse índice atinja o limite seu limitante r_{max}, uma perturbação é aplicada no centro do *cluster*, a fim de diversificar o espaço de busca e evitar ótimos locais (OLIVEIRA; CHAVES; LORENA, 2013).

Por último, uma busca local (LS) é realizada para cada área promissora da busca. Nesta etapa pode ser utilizada qualquer heurística de busca local para tentar encontrar a melhor solução dentro do *cluster* (OLIVEIRA; CHAVES; LORENA, 2013).

A Figura 3 apresenta o fluxograma do modelo conceitual do CS.

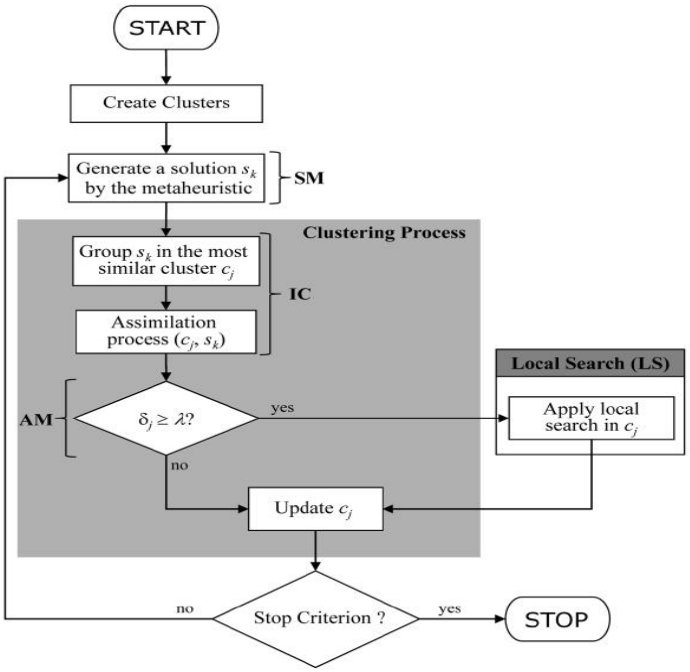

Figura 3 - Componentes do CS
Fonte: Oliveira, Chaves e Lorena (2013)

4. O ALGORITMO CS PROPOSTO

O *Clustering Search* proposto (Figura 4) utiliza o *Simulated Annealing* (SA) (KIRKPATRICK, GELATT E VECCHI, 1983) como meta-heurística geradora de soluções. Essa abordagem se baseia na simulação do processo termodinâmico de recozimento utilizado na fabricação de cristais e vidro (ZBIGNIEW; FOGEL, 2000). O SA é aplicado para encontrar soluções ótimas ou próximas do ótimo, permitindo explorar diferentes regiões do espaço de busca de maneira eficiente. Isso acontece pois é empregada uma probabilidade de se aceitar soluções piores durante sua execução, baseando-se na temperatura, que, quanto mais alta for, maior a probabilidade de se aceitar soluções piores e assim diversificar o espaço de busca (GOLDBARG; GOLDBARG; LUNA, 2016).

Os seguintes parâmetros são utilizados:

γ número máximo de *clusters*

λ volume máximo do *cluster*

r_{max} limite máximo do índice de ineficácia

T_0 temperatura inicial do SA

α taxa de resfriamento do SA

T_f temperatura final do SA

SA_{max} número de iterações por temperatura do SA

$Tempo_{max}$ tempo máximo de execução do CS

O algoritmo tem início com γ clusters vazios, cada um com um volume máximo de λ e um índice de ineficácia máximo de r_{max} (linha 1). Em seguida, é criada uma solução inicial s_o, que é considerada a melhor solução encontrada até o momento (s*) e como solução atual (s). Com os parâmetros definidos, inicia-se a execução do SA (linha 5) utilizando s_0 como base para geração de vizinhos. Gera-se uma solução vizinha s' a partir de s (linha 8), se s' for melhor do que s (linha 9), essa solução é definida como a nova solução atual (linha 10), e, caso essa nova solução atual seja melhor do que s*, ela é definida como a nova melhor solução (linha 12). Caso contrário, essa solução pode ser aceita com base em uma certa probabilidade relacionada à temperatura do SA (linha 15). Quanto maior a temperatura, maiores as chances de

uma solução pior ser escolhida como base para a geração de outros vizinhos. Esse processo é repetido até o limite SA_{max} ser atingido.

A temperatura do SA é decrementada (linha 18) de acordo com α, e, se existe algum *cluster* vazio, a solução s gerada pelo SA é atribuída a esse *cluster* (linha 20), caso não exista *clusters* vazios, a solução s é inserida no *cluster* que tem a menor distância de *Hamming* (HAMMING, 1950) entre ela e o centro do *cluster*. Além disso, o volume do *cluster* é incrementado, e caso a solução s seja melhor do que o centro do *cluster*, s é definido como o novo centro deste *cluster* (linha 22).

O *cluster* é considerado como promissor quando atinge o volume λ (linha 23), em seguida, seu volume é atualizado para 1 (linha 24), e avalia-se se deve ser explorado pela busca local a fim de melhorar o seu resultado (linha 25). A busca local é executada desde que o índice r_{max} não tenha sido atingido, caso contrário, uma perturbação é aplicada (consiste na geração de um novo vizinho) a este *cluster* a fim de diversificar o espaço de busca (linha 33).

Após a execução da busca local (linha 26), o centro do *cluster* é atualizado se a solução gerada for melhor do que a solução atualmente no centro (linha 28), e verifica-se também se esta solução é melhor do que a melhor solução atual s*, caso positivo, é definida como a nova melhor solução (linha 28).

Se a busca local não conseguir melhorar o centro do *cluster*, o índice de ineficácia desse *cluster* é incrementado (linha 30). Isso pode ser relevante para futuras iterações do algoritmo, pois indica que esse *cluster* pode não ser tão promissor para a solução geral.

Todo esse processo se repete até que o limite de tempo definido por $Tempo_{max}$ seja atingindo.

```
Entradas: γ, λ, r_max, T0, Tf, α, SA_max, Tempo_max, s_0
 v_i ← 0;  r_i ← 0;  s ← s_0;  s* ← s ;  Tempo ← 0;  index ← 0
ENQUANTO Tempo < Tempo_max FAÇA
    T ← T0
    ENQUANTO T > Tf FAÇA
        iteração ← 0
        ENQUANTO iteração < SAmax FAÇA
            s' ← N_1(s);   iteração ← iteração + 1
            SE f(s') < f(s) FAÇA
                s ← s'
                SE f(s) < f(s*) FAÇA
                    s* ←s
                FIM-SE
            SENÃO
                s ← s' com probabilidade e^(-(f(s')-f(s))/T)
            FIM-SE
        FIM-ENQUANTO
        T ← T × α
        SE index < γ FAÇA
            index ←index + 1 ;   v_index ← v_index + 1;   c_index ← s
        SENÃO
            i ← min_{∀i ∈ N: 1 < i < γ} {Hi};  v_i ← v_i + 1;   c_i ← melhor(c_i, s )
            SE v_i = λ ENTÃO
                v_i ← 1
                SE r_i < r_max FAÇA
                    s ← busca local( c_i )
                    SE f(s) < f(c_i ) ENTÃO
                        r_i ← 0;  c_i ← s; s* ← melhor(c_i, s*)
                    SENÃO
                        r_i ← r_i + 1
                    FIM-SE
                SENÃO
                    r_i ← 0; c_i ← N_1(c_i)
                FIM-SE
            FIM-SE
    FIM-ENQUANTO
FIM-ENQUANTO
RETORNE s*
```

Figura 4 - Peseudo código do algorítmo proposto.

4.1. A CONSTRUÇÃO DE SOLUÇÕES

O algoritmo proposto tem como objetivo trabalhar sempre com soluções viáveis, garantindo que as restrições de precedência e disponibilidade de recursos não sejam violadas durante a construção de uma solução ou a geração de um vizinho.

Ao adicionar uma tarefa à solução, é necessário respeitar a ordem de precedência, ou seja, a tarefa só pode ser selecionada após todas as suas predecessoras terem sido adicionadas. A inserção de uma atividade na solução ocorre em duas etapas:

1. Definição do início da execução da tarefa:

Nessa etapa, é determinado o momento a partir do qual a atividade pode começar a ser executada. Uma atividade só pode iniciar quando todas as suas predecessoras já foram concluídas. Portanto, o tempo de início da atividade será a partir do momento em que sua última predecessora terminou a execução.

2. O ajuste por disponibilidade de recursos:

Após determinar o momento de início da atividade, é necessário verificar se ela respeita as restrições de disponibilidade de recursos. Para isso, examina-se cada intervalo de tempo entre o início e o fim da atividade, a fim de identificar se há algum momento em que o número de recursos necessários para a execução da atividade excede o limite disponível. Caso isso aconteça, o início da tarefa é ajustado para o primeiro ponto em que nenhum limite de recurso é violado.

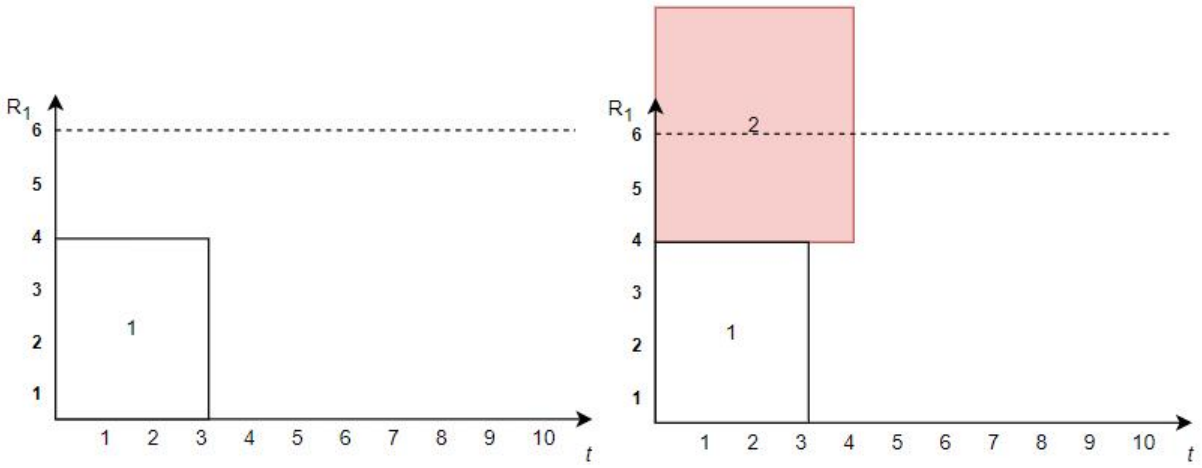

Figura 5 - Exemplo de inserção de atividades

A Figura 5 mostra a inserção de 3 atividades do projeto apresentado na Figura 1 na ordem {0, 1, 2}, a atividade 0 demarca o inicio do projeto, e tem duração e custo zero,

logo não é considerada na figura; ao inserir a atividade 1, o seu inicio é definido do ponto onde sua última predecessora (atividade 0) terminou, ou seja, no ponto 0. Em seguida, a atividade 2 é selecionada para entrar na solução, e seu ponto de inicio começa no momento onde sua última predecessora (atividade 0) termina, ou seja, o ponto 0. No entanto, ao inserir a atividade 2 no ponto 0, o número de recursos utilizados viola o limite de recursos disponíveis. Nesse momento, inicia-se a etapa 2, na qual verifica-se se a atividade inserida respeita as restrições de disponibilidade de recursos. Para realizar essa verificação, é examinado cada intervalo de tempo entre o início e o fim da atividade, buscando identificar se há algum momento em que o número de recursos necessários para a execução da atividade seja excedido. Se isso ocorrer em algum momento específico, o início da tarefa é ajustado para o primeiro ponto em que nenhum limite de recurso é violado, como mostrado na Figura 6.

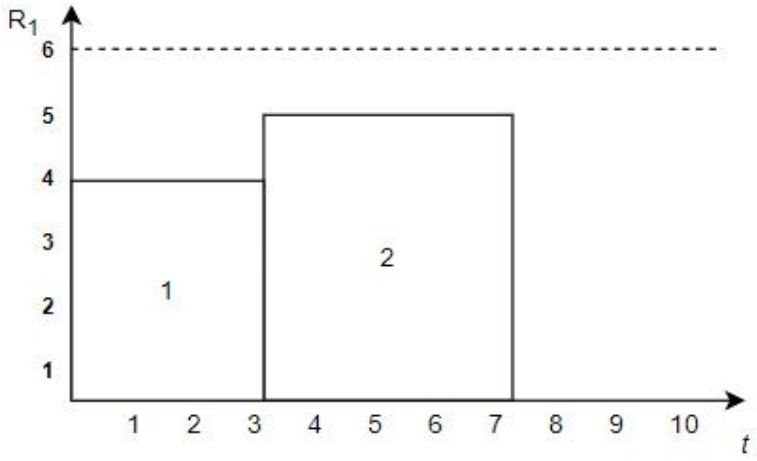

Figura 6. Exemplo de correção da posição da atividade.

4.2. A SOLUÇÃO INICIAL

A construção da solução inicial consiste em inserir as atividades na solução em ordem de precedência, utilizando as técnicas descritas na sessão 4.1 para garantir que todas as restrições estão sendo respeitadas.

4.3. GERAÇÃO DE SOLUÇÕES VIZINHAS

A estratégia de geração de vizinhança consiste em selecionar uma posição de inserção aleatória e realizar a troca dessa posição com a posição adjacente

posterior a ela. Antes da realização da troca, é verificado se a atividade anterior é predecessora da atividade posterior, e, caso não seja, a troca é realizada, se não uma nova posição aleatória é sorteada. Em seguida, uma vez que a troca de posição necessita de um ajuste em toda a solução, todas as tarefas são reinseridas na solução a partir da posição aleatória selecionada utilizando o método descrito na seção 4.1.

4.4. O CÁLCULO DE SIMILARIDADE

Para calcular a similaridade entre duas soluções é utilizado a distância de *Hamming* (HAMMING, 1950). Que consiste em analisar o vetor de ordem se inserção de cada solução e contabilizar quantas posições são diferentes uma da outra.

A Figura 5, apresenta a distância de *Hamming* entre duas soluções S1 e S2. A diferença entre as duas soluções está na posição 3 e 4, logo a distância de *Hamming* para essas duas soluções é igual a dois.

S_1	Sequência	1	2	3	4	5	6
	Atividade	2	1	4	3	5	6
S2	Sequência	1	2	3	4	5	6
	Atividade	2	1	3	4	5	6

Figura 7. Distância de Hamming entre duas soluções

4.5. BUSCA LOCAL

Na busca local foi utilizada uma estratégia de melhor melhora (HANSEN, P.; MLADENOVIĆ, N, 2006). Troca-se cada atividade com todas as posições da solução, armazenando aquela que minimiza mais o resultado. A cada iteração, a melhor solução encontrada é selecionada para outra rodada de trocas, repetindo esse processo até que todas as trocas sejam feitas sem observar qualquer melhoria adicional. Nesse momento, a execução é finalizada.

No cálculo da Função Objetivo (FO), emprega-se uma função com penalização para levar em conta a violação da precedência das atividades. Uma vez que todas as trocas serão efetuadas, algumas delas podem resultar na quebra da ordem de

precedência entre as atividade e, neste caso, aplica-se uma função com penalização para identificar essas violações durante o processo de otimização. Essa penalização consiste em somar a FO um valor consideravelmente grande.

5. TESTES COMPUTACIONAIS

Para avaliar a eficiência do CS aplicado ao RCPSP, foram selecionadas aleatoriamente seis instâncias de cada um dos três grupos de instâncias da biblioteca PSPLIB[1]: J30, J60 e J120, que possuem respectivamente 30, 60 e 120 atividades, cada uma delas com quatro tipos de recursos renováveis (sendo que a quantidade de cada tipo de recursos varia para cada instância), possuem também, a relação de ordem de precedência, a duração e o custo de recursos para cada atividade.

Durante a execução dos testes, o tempo de processamento foi limitado a 5 segundos para as instâncias do grupo J30, 15 minutos para as instâncias do grupo J60 e 40 minutos para as instâncias do grupo J120. Cada instância foi executada 5 vezes.

O algoritmo foi implementado na linguagem C++ e compilado pelo *GNU GCC Compiler* versão 9.4.0. A execução foi feita em um computador equipado com um processador *Intel Core i7* 11390H de 3.4 GHz, 16 GB de memória RAM e sistema operacional *Windows 10* de 64 bits.

5.1. A DEFINIÇÃO DOS PARÂMETROS

Os parâmetros foram definidos de acordo com o proposto por Bissoli, Zuffery e Amaral (2021) para o problema *Flexible Job Shop Scheduling Problem*, tendo em vista a sua semelhança com o problema RCPSP.

O parâmetro SA_{max} também foi definido de acordo com o proposto em Bissoli, Zuffery e Amaral (2021), que define um número I de iterações com base nas características de cada instância: $\mathrm{I} = 2 \times \mathrm{r} \times (\mathrm{n} - 2)$ onde r é o número de recursos e n o numero de atividades de cada instância.

1. Disponível em: https://www.om-db.wi.tum.de/psplib/main.html

Dessa forma os parâmetros ficaram definidos da seguinte forma:

Parâmetro	Valor
T_0	1000
T_f	0,001
SA_{max}	I
α	0,998
γ	20
λ	5
r_{max}	2

Tabela 1 - Valores dos parâmetros do CS

6. RESULTADOS

A seguir, será apresentado os resultados computacionais do CS aplicado ao RCPSP. O objetivo é analisar a sua robustez, e compará-lo com as demais abordagens encontradas na literatura.

6.1. RESULTADOS COMPUTACIONAIS

A Tabela 2 apresenta o resumo dos resultados obtidos para cara grupo de instâncias. Nela é apresentado a melhor função objetivo obtida, a média aritmética do resultado das cinco execuções, o desvio em relação à melhor função objetivo encontrada calculado por $\text{Desvio}(\%) = \frac{\text{Média} - \text{MelhorFO}}{\text{MelhorFO}} \times 100,$ e tempo médio para encontrar a melhor solução.

	Instância	Melhor FO	Média	Desvio (%)	Tempo médio (s)
J30	J301_1	43	43	0	0,172
	J301_2	47	47	0	0,426
	J3024_5	51	51	0	0,000
	J3024_6	56	56	0	0,000
	J3048_9	59	54	0	0,000
	J3048_10	54	59	0	0,000
J60	J601_1	77	77	0	0,150
	J601_2	68	68	0	444,382
	J6024_4	78	78	0	0,000
	J6024_5	76	76	0	0,000
	J6048_9	82	70	0	0,000
	J6048_10	70	82	0	0,000
J120	X1_1	115	115,2	0,17	676,923
	X1_2	119	120	0,84	979,690
	X48_5	118	119,4	1,19	1329,829
	X48_6	116	116,4	0,34	1360,577
	X60_9	101	101	0	290,234
	X60_10	95	95,83	0,88	975,246

Tabela 2 - Resultados obtidos pelo CS para as instâncias J30, J60 e J120 selecionadas.

Pode-se observar que os resultados para as instâncias J30 e J60 foram consistentes apresentando 0 de desvio em todas as execuções. Para as instâncias J120 o pior resultado foi para a instância X48_5 com desvio de 1,19% dentro das cinco execuções. No geral, obteve-se um desvio médio de 0,19%.

Em relação ao tempo computacional, para a instância J30 o CS apresentou uma baixa variação, já para as instâncias J60 e J120 podemos observar que o tempo pode variar bastante dependendo das características de cada instância.

6.2. COMPARAÇÃO COM A LITERATURA

A Tabela 3 apresenta os resultados em relação aos melhores resultados encontrados na literatura, sendo que os resultados para as instâncias J30 representam o valor ótimo. Os valores foram obtidos através site da biblioteca PSPLIB[2]. O desvio foi calculado por $\text{Desvio}(\%) = \frac{\text{Literatura} - \text{Melhor FO}}{\text{Melhor FO}} \times 100.$

	Instância	Literatura	CS	Desvio (%)
J30	J301_1	***43**	***43**	0
	J301_2	***47**	***47**	0
	J3024_5	***51**	***51**	0
	J3024_6	***56**	***56**	0
	J3048_9	***59**	***59**	0
	J3048_10	***54**	***54**	0
J60	J601_1	***77**	***77**	0
	J601_2	***68**	***68**	0
	J6024_4	***78**	***78**	0
	J6024_5	***76**	***76**	0
	J6048_9	***82**	***82**	0
	J6048_10	***70**	***70**	0
J120	X1_1	**105**	115	9,52
	X1_2	**109**	119	9,17
	X48_5	**110**	118	7,27
	X48_6	**105**	116	10,48
	X60_9	***101**	***101**	0
	X60_10	**89**	95	6,74

Tabela 3 - Comparação dos resultados obtidos pelo CS com os melhores resultados da literatura.

Os resultados em negrito representam o melhor resultado conhecido, os resultados com asterisco (*) indicam um empate entre os resultados obtidos pelo método proposto e os resultados da literatura.

Analisando os resultados podemos perceber que o CS encontrou todos os melhores resultados para as instâncias J30 e J60 selecionadas, enquanto que para as instâncias J120 apenas para a instância X60_9 o melhor resultado foi encontrado. No geral, obteve-se um desvio médio de 2,4%.

2. Disponível em: https://www.om-db.wi.tum.de/psplib/getdata_sm.html

7. CONCLUSÃO

A aplicação do *Clustering Search* (CS) para a resolução do problema de escalonamento de projetos com restrições de recursos (RCPSP) mostrou-se promissora. Os resultados obtidos demonstraram a robustez e consistência do CS, especialmente para as instâncias do grupo J30 e J60. Em todas as execuções, o CS atingiu todos os melhores resultados reportados na literatura, indicando sua capacidade de encontrar soluções de alta qualidade.

Além disso, o CS revelou um potencial promissor para lidar com as instâncias do grupo J120. Embora não tenha alcançado o resultado ótimo para todas as instâncias testadas, conseguiu atingir o resultado da literatura para a instância X60_9 e mostrou-se próximo das soluções para as demais instâncias avaliadas. Essa capacidade de aproximação sugere que o CS pode ser uma abordagem viável para resolver as instâncias mais complexas do RCPSP.

No que diz respeito ao tempo computacional, observou-se uma consistência nos resultados apresentados para as instâncias do grupo J30. O CS mostrou-se eficiente e capaz de produzir soluções de qualidade dentro de limites de tempo razoáveis. No entanto, para as instâncias dos grupos J60 e J120, o tempo computacional pode variar consideravelmente, dependendo das características específicas de cada instância. Essa variabilidade deve ser considerada ao selecionar o método do CS para resolver instâncias com mais de 30 atividades.

Como resultado, o Clustering Search demonstrou ser uma abordagem consistente e eficaz para a resolução do problema de escalonamento de projetos com restrições de recursos. Com bons resultados para as instâncias do grupo J30 e J60, além de um potencial encorajador para as instâncias do grupo J120, o CS mostra-se como uma alternativa valiosa na busca por soluções ótimas ou aproximadas. No entanto, a análise cuidadosa do tempo computacional é essencial ao aplicar o CS, especialmente em problemas com maior número de atividades.

Olhando para o futuro, como trabalhos a serem desenvolvidos e possíveis melhorias, investir em uma busca local mais eficiente pode potencialmente aprimorar ainda mais os resultados obtidos.

REFERÊNCIAS

BALLESTÍN, F.; VALLS, V.; QUINTANILLA, S. Scheduling Projects with Limited Number of Preemptions. **Computers & Operations Research,** vol. 36 n. 11 p. 2913-925, 2009.

BISSOLI, DAYAN C.; ZUFFEREY, NICOLAS; AMARAL, ANDRÉ RS. Lexicographic optimization-based clustering search metaheuristic for the multiobjective flexible job shop scheduling problem. **International Transactions in Operational Research**, vol. 28, n. 5, p. 2733-2758, 2021.

CHEN, WANG; SHI, YAN-JUN; TENG, HONG-FEI; LAN, XIAO-PING; HU, LI-CHEN. An efficient hybrid algorithm for resource-constrained project scheduling. **Information Sciences**, vol. 180, n. 6, p. 1031-1039, 2010.

DE ABREU, VICTOR HUGO SOUZA ; GONZÁLEZ, PEDRO HENRIQUE ; MAURI, GERALDO REGIS ; RIBEIRO, GLAYDSTON MATTOS ; ORRICO, ROMULO DANTE ; CAMPOS JÚNIOR, NILO FLAVIO ROSA ; ABRAMIDES, CARLOS ALBERTO. Network sensor location problem with monitored lanes: Branch-and-cut and clustering search solution techniques**. Computers & Industrial Engineering**, vol. 150, p. 106827, 2020.

ROSA, RODRIGO DE ALVARENGA; MACHADO, ANDRÉ MANHÃES; RIBEIRO, GLAYDSTON MATTOS; MAURI, GERALDO REGIS. A mathematical model and a Clustering Search metaheuristic for planning the helicopter transportation of employees to the production platforms of oil and gas. **Computers & Industrial Engineering**, vol. 101, p. 303-312, 2016.

ELSAYED, SABER; SARKER, RUHUL; RAY, TAPABRATA; COELLO, CARLOS COELLO. Consolidated optimization algorithm for resource-constrained project scheduling problems. **Information Sciences**, vol. 418, p. 346-362, 2017.

GOLDBARG, M. C.; GOLDBARG, E. G.; LUNA, H. P. L. Otimização combinatória e meta-heurísticas: Algoritmos e aplicações. **Rio de Janeiro: Elsevier**, 2016. ISBN 978-85-352-7812-5.

HAMMING, R. W. Error detecting and error correcting codes. **Bell System Technical Journal**, vol. 26, p. 147-160, 1950.

HANSEN, P.; MLADENOVIĆ, N. First vs. best improvement: An empirical study. **Discrete Applied Mathematics**, v. 154, n. 5, p. 802 – 817, 2006.

LABORIE, P. Algorithms for Propagating Resource Constraints in AI Planning and Scheduling: Existing Approaches and New Results. **Elsevier ScienceDirect Journals**. vol. 143 n. 2 p. 151-88, 2003.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. **Optimization by simulated annealing**. Science, v. 220, n. 4598, p. 671- 680, 1983.

OLIVEIRA, ALEXANDRE C. M DE; CHAVES, ANTONIO A.; LORENA, LUIZ A. N. Clustering search. **Pesquisa operacional**, vol. 33, n.1, p. 105-121, 2013.

PARASKEVOPOULOS, DIMITRIS C.; TARANTILIS, CHRISTOS D.; IOANNOU, GEORGE. Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. **Expert Systems with Applications**, vol. 39, n. 4, p. 3983-3994, 2012.

PELLERIN, ROBERT; PERRIER, NATHALIE; BERTHAUT, FRANÇOIS. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. **European Journal of Operational Research**, vol. 280, n. 2, p. 395-416, 2020.

PROON, SEPEHR; JIN, MINGZHOU. A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem. **Naval Research Logistics (NRL)**, vol. 58, n. 2, p. 73-82, 2011.

ZBIGNIEW, M.; FOGEL, D.B. **How to Solve it: Modern Heuristics**. 2 ed. Springer-Verlag, 2000.