XXI ENMC
ENCONTRO NACIONAL DE MODELAGEM COMPUTACIONAL
IX ECTM
ENCONTRO DE CIÊNCIA E TECNOLOGIA DE MATERIAIS

08 a 11 de Outubro de 2018
Instituto Federal Fluminense
Búzios - RJ

PROPOSTA DE UM MÉTODO PARA O PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS

Leonardo Martins Freiria[1] – freiria.leonardo@gmail.com
Thiago Alves Queiroz[1] – taq@ufg.com
[1] Universidade Federal de Goiás - Regional Catalão, 75704-020, Catalão-GO, Brasil.

Resumo. *O problema de programação de projetos com restrição de recursos, que é considerado um problema NP-díficil, é resolvido neste trabalho por uma heurítica de busca do tipo tabu. A heurística considera movimentos de troca, inserção e inserção de sequências de atividades. As soluções da heurística proposta são comparadas com resultados da literatura, levando em consideração a minimização do makespan e o tempo gasto para obter a solução. Apesar da heurística considerar três movimentos para obter novas soluções, ela conseguiu encontrar a melhor solução conhecida de várias instâncias, comprovando a eficácia da busca tabu proposta.*

Palavras-chave: *Busca tabu, Problema de escalonamento de projetos, Heurística, Makespan.*

1. INTRODUÇÃO

O Problema de Programação de Projetos com Restrição de Recursos (*Resource Constrained Project Scheduling Problem* – RCPSP) consiste em escalonar um conjunto de atividades, que possuem dependência de precedência e de recursos, em um dado horizonte de tempo (Bruni, 2017). As atividades a serem escalonadas possuem tempo de processamento e consomem uma determinada quantidade de cada recurso disponível. O RCPSP busca determinar o tempo de início das atividades, obedecendo as restrições intrínsecas do problema e objetivando a minimização da duração total do projeto, conhecido por *makespan*. O RCPSP é um problema de otimização combinatória, classificado como NP-difícil (Blazewicz,1983), de forma que não se conhece e não se espera métodos exatos de tempo polinomial.

A proposta de Naber (2017) está baseada em um algoritmo heurístico para resolver uma variante do RCPSP em que as quantidades dos recursos são flexíveis. Para tanto, os autores consideraram que as atividades são não-preemptivas, ou seja, uma vez iniciada a atividade, deve-se concluí-la sem interrupção; há relações de precedência, em que a atividade poderá iniciar somente após o término de todas as suas antecessoras; respeitar o consumo de recursos,

em que a quantidade de recursos consumida em cada horizonte de tempo não ultrapasse a quantidade disponível; e, há a utilização fixa de recursos pelas atividades.

Em Asta (2016) fez-se uma combinação entre o Monte-Carlo e um meta-heurística memética, que é uma variante dos algoritmos genéticos. Na abordagem são determinadas duas fases: construir e melhorar. Na primeira fase, cria-se uma sequência inicial válida. Em seguida, aplica-se um conjunto de movimentos de vizinhança buscando melhorar essa solução. Apesar da limitação dos métodos exatos para a obtenção rápida de uma solução ótima de instâncias médias e grandes, em Coelho & Vanhoucke (2018) foi implementado uma estratégia para gerar um limitante inferior em conjunto com um método exato do tipo *branch-and-bound*, sendo que o limitante inferior é atualizado dinamicamente no procedimento de busca, visando melhorar o tempo para obter uma solução ótima.

A partir disso, desenvolve-se uma heurística fundamentada na busca tabu para o RCPSP, a qual utiliza estruturas de memória para impedir soluções repetidas e movimentos, a fim de melhorar as soluções. Para a construção da heurística, busca-se devolver boas soluções em baixo tempo computacional, com o propósito de auxiliar os gestores na tomada de decisões rápidas e precisas. O artigo está dividido em cinco seções. A Seção 2 abrange a definição do problema, com a apresentação das restrições e do objetivo, bem como discute sobre a heurística desenvolvida. Os experimentos computacionais são mencionados na Seção 3, contemplando uma discussão e comparação de resultados com a literatura. Por fim, na Seção 4 há as conclusões obtidas e sugestões para trabalhos futuros.

2. PROBLEMA E MÉTODO DE SOLUÇÃO

Dumic (2018) ressalta a intensa procura de novos métodos de solução para o RCPSP, desde a sua definição na década de 1960. Esse problema de escalonamento com restrições de recurso é difícil do ponto de vista computacional, o que tem motivado a literatura desenvolver métodos de solução heurísticos. A próxima subseção descreve formalmente o RCPSP para, em seguida, descrever-se a proposta de uma heurística para tal problema.

2.1. DEFINIÇÃO DO PROBLEMA

O RCPSP é constituído por um conjunto M de atividades $\{0, 1, 2, ..., n, n+1\}$ e por um conjunto R de recursos $\{1, 2, ..., m\}$, sendo que para cada atividade $i \in M$ tem-se os seguintes conjuntos/parâmetros: ***pred***, contém as relações de precedência (i, j) entre as atividades; $\boldsymbol{d_i}$, representa o tempo de processamento da atividade i; $\boldsymbol{C_{ik}}$, que é a quantidade do recurso k consumida pela atividade i; e, $\mathbf{R_k}$, representa a quantidade disponível do recurso renovável k. Vale destacar que as atividades 0 e $n+1$ são para o controle, chamada de atividades fictícias, de forma que não consomem recursos e possuem tempo de processamento nulo (Brucker, 1998).

Buscando a minimizar o tempo de duração do projeto, que é denominado por *makespan*, tem-se que atividades podem ser executadas em paralelo, isto é, simultaneamente, desde que a relações de precedência sejam satisfeitas e que o consumo de recursos por essas atividades não ultrapasse a quantidade disponível dos recursos. A partir do início de uma atividade, a mesma deve ser executado por completo, sendo vedada qualquer preempção de atividade. Ainda, ressalta-se que não são considerados tempo de preparação entre as atividades. A Tabela 1 traz um exemplo de instância para o RCPSP que contém as relações de precedência ilustradas na Figura 1.

Tabela 1 - Exemplo de instância para o RCPSP, com 12 atividades e 2 recursos renováveis.

Atividade i	d_i	$R_{i,0}$	$R_{i,1}$	Sucessoras
0	0	0	0	{1,2,3,4}
1	6	2	1	{10}
2	1	1	0	{5,6}
3	1	3	1	{7}
4	2	2	0	{8}
5	3	1	1	{9}
6	5	2	1	{10}
7	6	3	0	{11}
8	3	1	2	{11}
9	2	1	2	{10}
10	4	1	1	{11}
11	0	0	0	
Disponível		**7**	**4**	

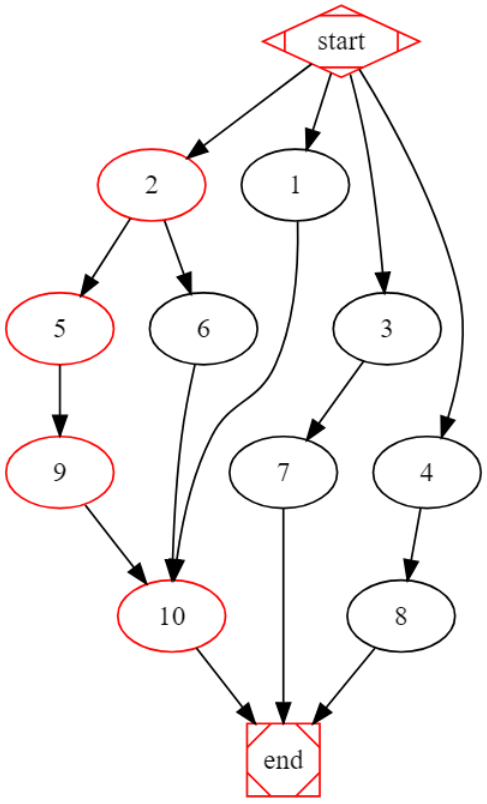


Figura 1 - Grafo de precedência para o conjunto de 12 atividades da Tabela 1.

2.2. HEURÍSTICA PARA O RCPSP

A heurística de busca tabu (do inglês, *tabu search*) parte de uma solução inicial, em que operações (isto é, movimentos) são realizadas entre as atividades, buscando gerar soluções vizinhas à solução inicial e realizando a substituição da solução corrente por uma solução vizinha melhor gerada a partir dos movimentos (Glover, 1997). O aspecto mais vantajoso da busca tabu deve-se à utilização de uma memória adaptativa, ou seja, as soluções já avaliadas

são armazenadas em uma lista tabu. Como isso, busca-se evitar soluções repetidas, uma vez que os movimentos realizados para alcançar tais soluções são também armazenados, possibilitando a saída de ótimos locais.

De acordo com Pinedo (2008), heurísticas fundamentadas em busca local devem obter um escalonamento com todas as atividades do projeto. Logo, para a determinação da solução inicial do RCPSP, utiliza-se uma ordenação topológica (Bukata *et al.*, 2015). Por meio dessa ordenação, prioriza-se em satisfazer as relações de precedência entre as atividades. Neste estágio não são determinados os tempos de início das atividades e as possibilidades de atividades que podem ser escalonadas em paralelo.

Conforme ilustrado no pseudocódigo da Figura 2, finalizada a geração da solução inicial para o RCPSP, cria-se a lista tabu para armazenar soluções e o seu respectivo custo. Vale ressaltar que caso a solução permaneça na mesma solução durante cerca de 1% (representado por Θ) da quantidade de iterações MAX, que é de 100.000, a heurística assume tal solução como a final, considerando que a contagem dessas iterações é válida apenas para os movimentos que satisfazem as relações de precedência. Observa-se que ao passo que a solução é melhorada, a contagem de iterações sem melhora é reiniciada.

```
                Algoritmo Tabu Search- TS
1:  → Gera uma solução inicial X
2:  → Criar Lista Tabu- TB
3:  → TB ← ∅
4:  → cont=0
5:  enquanto (não atingiu MAX iterações) faça
6:     α ←  Faça movimento (X)
7:     se (α  ∈  TB) então
8:        Volte ao ínicio do laço
9:     fim se
10:    cont++
11:    Y  ← Executar α na solução X
12:    se (f(Y)  < f(X) ) então
13:       X ← Y
14:       cont=0
15:    fim se
16:    Adicionar α na lista tabu- TB no local do movimento mais antigo
17:    → atribuir "idade "0 para α
18:    → Guardar o escalonamento e custo da solução
19:    se (cont atingir Θ iterações)  então
20:       break
21:    fim se
22: fim enquanto
    Retorna a solução em X
```

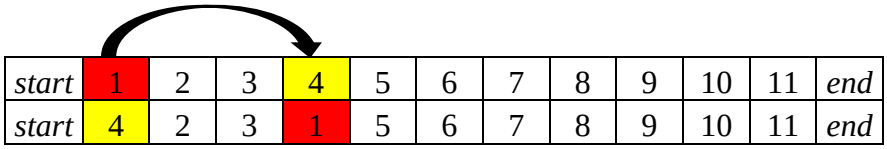
Figura 2 - Pseudocódigo da busca tabu desenvolvida para o RCPSP.

As atividades a serem escalonadas, devido as relações de precedência, podem ser organizadas em um grafo direcionado, em que os nós são as próprias atividades e as arestas determinam as relações de precedência, conforme ilustrado na Figura 1. A atividade que inicia o processamento e a outra que finaliza são atividades fictícias, utilizadas unicamente para controle, sem qualquer consumo de recursos ou duração, sendo as atividades *start* e *end* no presente exemplo. Para a determinação da solução inicial, calculam-se as atividades predecessoras, sendo que a atividade corrente estará disponível para o escalonamento quando as suas predecessoras já estiverem escalonadas. Conforme o exemplo na Figura 1, a primeira atividade a ser escalonada é a *start*, pois é a única atividade sem qualquer predecessora. Após ser escalonada, remove-se os arcos (*start*, 1), (*start*, 2), (*start*, 3) e (*start*, 4). Desta forma, as atividades 1, 2, 3 e 4 não possuem mais dependentes e, consequentemente, podem ser

escalonadas após a atividade 0. Esses mesmos passos são repetidos para as atividades seguintes até atingir a atividade *end.* Assim, o escalonamento é *start*-1-2-3-4-5-6-7-8-9-10-*end.*

Para obter a solução inicial, criam-se também conjuntos de ordenação topológica, de forma que se agrupam atividades com a mesma quantidade de predecessoras, além de determinar os possíveis *caminhos críticos*, isto é, atividades que estão diretamente ligadas a minimização do *makespan*. Na Figura 1, tais caminhos correspondem aos nós (e seus arcos) na cor vermelha. Na Figura 2 são formados seis níveis topológicos: 1º nível com (*start*), 2º nível com (1, 2, 3, 4), 3º nível com (5, 6, 7, 8), 4º nível com (9), 5º nível com (10) e 6º nível com (11). Com os níveis da ordenação topológica, tem-se apenas as relações de precedência sendo respeitadas, tornando-se necessário avaliar a disponibilidade dos recursos em cada instante de tempo. Prioriza-se escalonar em paralelo as atividades do mesmo nível de forma a reduzir o tempo de conclusão do projeto, respeitando-se a limitação dos recursos disponíveis.

As soluções vizinhas da solução corrente são obtidas pela realização de movimentos de troca e inserção, analisados tanto nos conjuntos topológicos quanto na ordenação topológica. Como os movimentos não podem violar as relações de precedência, pode ser necessário a criação de outros níveis topológicos. Os movimentos desenvolvidos para melhoria da solução, obtendo soluções vizinhas, baseiam-se na **troca** de posições entre duas atividades, **inserção** de uma atividade antes da outra atividade e **inserção de uma sequência** de atividades antes de outra atividade. Na Figura 3 exemplifica-se os movimentos de troca (em 3a), inserção de uma atividade (em 3b) e inserção de uma sequência (em 3c) respeitando as precedências da Figura 1. Após a efetivação do movimento entre as atividades, faz-se a avaliação dos recursos e atualização das matrizes da solução. O movimento realizado é salvo na lista tabu e a heurística seleciona um novo movimento, até atingir um número MAX de iterações ou se houver uma sequência de iterações sem melhoria da solução corrente. Caso o movimento resulte em uma solução pior do que a atual, tal movimento não é efetuado.

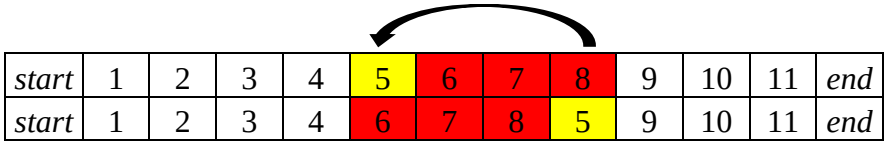


start	1	2	3	4	5	6	7	8	9	10	11	*end*
start	4	2	3	1	5	6	7	8	9	10	11	*end*

(a) Exemplo de troca entre duas atividades.

start	1	2	3	4	5	6	7	8	9	10	11	*end*
start	1	2	3	4	6	7	5	8	9	10	11	*end*

(b) Exemplo de inserção de uma atividade.

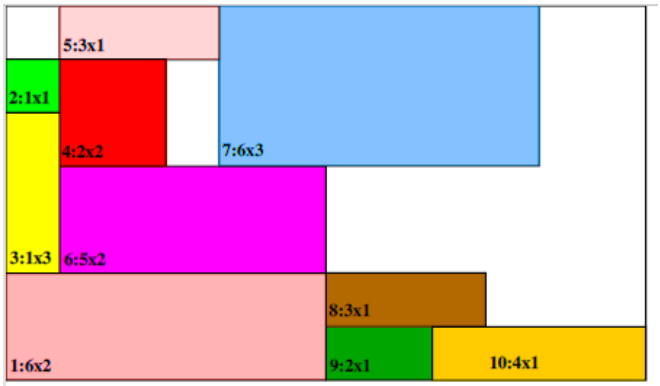


start	1	2	3	4	5	6	7	8	9	10	11	*end*
start	1	2	3	4	6	7	8	5	9	10	11	*end*

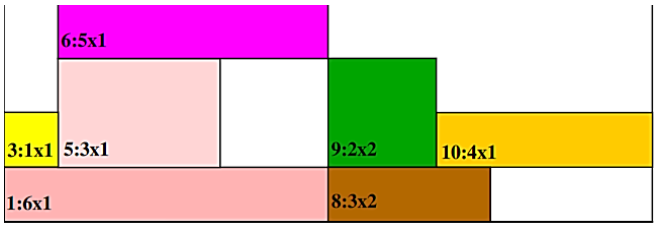
(c) Exemplo de inserção de uma sequência de atividades.

Figura 3 - Exemplo dos movimentos considerados na busca tabu.

A Figura 4 ilustra uma solução obtida pela heurística de busca tabu para o exemplo de instância apresentado na Tabela 1. A melhor solução obtida pela heurística tem o *makespan* igual a 12 e com a sequência de escalonamento *start*-1-2-3-4-5-6-7-8-9-10-*end*. A figura exemplificada permite observar que as relações de precedência foram respeitadas, bem como a restrição de recurso.



(a) Escalonamento das atividades respeitando o primeiro recurso.



(b) Escalonamento das atividades respeitando o segundo recurso.

Figura 4 - Solução encontrada pela busca tabu para a instância na Tabela 1.

3. TESTES NUMÉRICOS

A heurística de busca tabu discutida na seção anterior foi codificada na linguagem C++. Os testes foram realizados sobre as instâncias da PSPLIB (*Project Scheduling Problem Library*). O computador adotado para os testes possui sistema operacional Ubuntu 16.04 LTS. As instâncias da PSPLIB escolhidas são do conjunto J30 e possuem 32 atividades a serem escalonadas, sendo duas destas atividades utilizadas para controle, isto é, atividades fictícias. Na Tabela 2 estão os resultados das 480 instâncias do conjunto J30.

Tabela 2 - Resultados da busca tabu para as 480 instâncias do conjunto J30.

Grupo	Solução da Literatura	Solução da busca tabu	Tempo da busca tabu	Desvio
J301	49,3	53,5	0,87	8,52%
J302	47,1	49,8	1,32	5,73%
J303	60	61,4	0,86	2,33%
J304	51,4	51,4	0,23	0,00%
J305	67,6	74,3	2,01	9,91%
J306	50,1	55,8	1,37	11,38%
J307	46,5	48,7	0,52	4,73%
J308	49,9	49,9	0,52	0,00%
J309	74,8	84,6	1,88	13,10%
J310	49,6	54,7	1,04	10,28%
J311	55	59,5	2,06	8,18%

Tabela 2 - Resultados da busca tabu para as 480 instâncias do conjunto J30 (continuação).

Grupo	Solução da Literatura	Solução da busca tabu	Tempo da busca tabu	Gap
J312	49,2	49,2	0,562	0,00%
J313	71,7	82	2,208	14,37%
J314	50,9	56,5	1,359	11,00%
J315	53	55	0,829	3,77%
J316	45,7	45,7	0,569	0,00%
J317	58,3	60,6	0,649	3,95%
J318	54,4	57,5	0,637	5,70%
J319	51,4	53,8	1,072	4,67%
J320	50,2	50,2	0,470	0,00%
J321	68,5	75,7	1,742	10,51%
J322	54,2	59,2	1,516	9,23%
J323	55,9	59,2	1,288	5,90%
J324	51,8	51,8	0,502	0,00%
J325	76,1	87,1	2,185	14,46%
J326	56	59,1	0,904	5,54%
J327	56,5	61,3	1,372	8,50%
J328	56,5	56,5	0,286	0,00%
J329	87,2	98,1	2,200	12,50%
J330	55,2	62	1,630	12,32%
J331	54,3	57,1	1,311	5,16%
J332	54,9	54,9	0,320	0,00%
J333	60,6	63,1	1,062	4,13%
J334	58,6	60,2	1,303	2,73%
J335	58	61,2	1,032	5,52%
J336	57,7	57,7	0,023	0,00%
J337	76,7	85,2	1,689	11,08%
J338	60,3	64,7	1,325	7,30%
J339	58,7	59,9	0,366	2,04%
J340	56,4	56,4	0,262	0,00%
J341	91,1	104,6	2,232	14,82%
J342	61,8	66,3	2,000	7,28%
J343	57,5	62,1	1,238	8,00%
J344	54,1	54,1	0,281	0,00%
J345	96,5	108,1	2,233	12,02%
J346	59,2	64,9	1,357	9,63%
J347	56	61,1	2,028	9,11%
J348	55,2	55,2	0,306	0,00%

A Tabela 2 traz os resultados da heurística de busca tabu agrupados em 10 instâncias por grupo, sendo dados o nome de cada grupo, o valor médio da função objetivo retornado pela literatura, o valor médio da função objetivo retornado pela busca tabu, o valor médio do tempo gasto pela busca tabu (em segundos) e o desvio relativo (em porcentagem) entre o valor da solução da busca tabu e o valor da literatura. A Figura 5 representa o valor do desvio relativo para as 480 instâncias.

O desvio relativo na Tabela 2 é inferior a 15% para todos os 48 grupos formados, sendo que a heurística de busca tabu foi capaz de encontrar a mesma solução reportada na literatura para 12 dos grupos por completo, isto é, com desvio igual a 0% para todas as instâncias do grupo. Fazendo uma análisa por instância, o resultado é ainda melhor, pois a busca tabu foi capaz de encontrar a solução com desvio igual a 0% para cerca de 50% das instâncias do conjunto J30. Considerando um desvio inferior a 5%, o total de instâncias sobe para 66% das 480 utilizadas, representando 316 instâncias, o que é um resultado promissor para a heurística.

Ainda observando os resultados da Tabela 2, nota-se que para 19% das instâncias, a solução teve um desvio superior superior a 10%, sendo preciso desenvolver outros movimentos para que a solução não estacione em um ótimo local. Com relação ao tempo gasto para resolver as instâncias, nota-se que o processamento mais longo para os grupos foi de cerca de 3 segundos, mostrando que a heurística permite um auxílio rápido para tomada de decisões nas empresas.

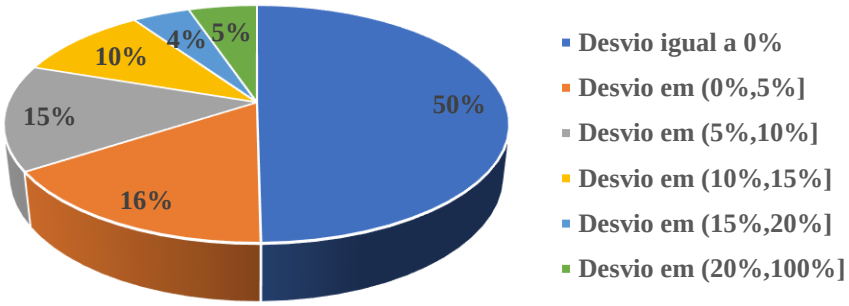


Figura 5 - Porcentagem de instâncias conforme o valor do desvio relativo na Tabela 2.

4. CONCLUSÕES

Os problemas de programação da produção são encontrados em atividades cotidianas, principalmente problemas de escalonamento de projetos com restrição de recursos. Este problema, por ser classificado como NP-difícil, tem sido geralmente resolvido por heurísticas, como é o caso do presente trabalho. A proposta de uma heurística de busca tabu permite que soluções sejam obtidas rapidamente, permitindo auxiliar o tomador de decisão que trabalha sobre tal problema.

A heurística desenvolvida contém movimentos de troca, inserção de uma atividade e inserção de uma sequência de atividades, possibilitando obter a melhor solução reportada na literatura para várias instâncias do problema em estudo. Como trabalho futuro, pretende-se aprimorar os movimentos, com a criação de novos movimentos para evitar ótimos locais, além de considerar outras restrições para o problema, como o caso de janela de tempo e preempção para as atividades.

Agradecimentos

Os autores agradecem o apoio financeiro recebido do Conselho Nacional de Desenvolvimento Tecnológico e Científico (processo nº 308312/2016-3).

REFERÊNCIAS

Asta, S., Karapetyan, D., Kheiri, A. e Özcan, E. (2016), Combining Monte-Carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem. *Information Sciences*, 373, 476-498.

Blazewicz, J., Lenstra, J.K. e Rinnooy Kan, A.H.G. (1983), Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5, 11-24.

Brucker, P., Knust, S., Schoo, A. e Thiele, O. (1998), A branch and bound algorithm for the Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research*, 107, 272-288.

Bruni, M.E., Pugliese, L.D.P., Beraldi, P. e Guerriero, F. (2017), An adjust able robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations. *Omega*, 71, 66-84.

Bukata, L., Sucha, P. e Hanzálek, Z. (2015), Solving the resource constrained project scheduling problem using the parallel tabu search designed for the CUDA plataforma, *Journal of Parallel and Distributed Computing*, 77, 58-68.

Coelho, J. e Vanhoucke, M. (2018), An exact composite lower bound strategy for the resource-constrained Project scheduling problem. *Computers & Operations Research*, 93, 135-150.

Dumic, M., Sisejkovic, D., Coric, R., Jakobovic, D. (2018), Evolviving priority rules for resource constrained project scheduling problem with genetic programming. *Future Generation Computer Systems*, 86, 211-221.

Glover, F. e Laguna, M. (1997), "*Tabu Search*", Kluwer Academic Publishers, Boston.

Naber, A. (2017), *Resource-constrained project scheduling with flexible resource profiles in continuous time. Computers & Operations Research*, 84, 33-45.

Pinedo, M. L. (2008), "*Scheduling: theory, algorithms, and systems*", 3ª ed., Springer, New York.

A PROPOSE OF A METHOD TO THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM

Abstract*. The Resource-Constrained Project Scheduling Problem (RCPSP), which is considered an NP-hard problem, is solved in this work by a search heuristic based on a tabu search. The tabu search considers the following movements related to activities: swap, insertion, and insertion of a sequence. The solutions found by means of the proposed heuristic are compared with the literature by taking into account the minimization of the makespan and the time spent to obtain the solution. Although the heuristic considers only three movements to obtain new solutions, it was possible to find out the best-known solution of several instances, proving the effectiveness of the proposed tabu search.*

Keywords: *Tabu search, Project Scheduling Problem, Heuristic, Makespan.*