UMA ABORDAGEM BASEADA EM *ITERATED LOCAL SEARCH* PARA O PROBLEMA DE ESCALONAMENTO EM PROJETOS COM RESTRIÇÃO EM RECURSOS E MÚLTIPLOS MODOS DE EXECUÇÃO

Gustavo Alves Fernandes
Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG
Av. Amazonas, 7675 - Nova Gameleira- CEP 30510-000 - Belo Horizonte, MG
gustavo.alfer@gmail.com

Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG
Av. Amazonas, 7675 - Nova Gameleira- CEP 30510-000 - Belo Horizonte, MG
sergio@dppg.cefetmg.br

Moacir Felizardo de França Filho
Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG
Av. Amazonas, 7675 - Nova Gameleira- CEP 30510-000 - Belo Horizonte, MG
franca@des.cefetmg.br

Resumo

Nesse artigo, considera-se o Problema não Preemptivo de Escalonamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução para a minimização do *makespan*. Para solucionar esse problema, foi proposta a metaheurística *Iterated Local Search* (ILS). O método de Descida de Primeira Melhora é a busca local para o ILS, porém, essa busca é realizada em dois estágios, busca por modos de execução e, após, busca por atividade. Os resultados obtidos, para um grupo de instâncias entre 3 classes de problemas da PSPLIB, mostraram que a abordagem proposta foi capaz de encontrar boas soluções em tempo aceitável.

Palavras-Chaves: Escalonamento de Projetos; Otimização Combinatória; Metaheurística; *Iterated Local Search*.

Abstract

In this paper we consider the nonpreemptive Multi-Mode Resource Constrained Project Scheduling Problem (MRCPSP). The goal is the minimization of the makespan. To solve this problem, we propose an Iterated Local Search (ILS) metaheuristic. The First Improvement Descent Method is the intensification phase used by ILS but, unlike conventional, the local search is performed in two stages, improvement by mode and improvement by activity. Results obtained for a group of instances among three classes of test problems by PSPLIB have shown that our algorithm was capable of find good solutions in acceptable time.

Keywords: Project scheduling; Combinatorial Optimization; Meta-heuristic; Iterated Local Search;

1. INTRODUÇÃO

O problema clássico de Escalonamento em Projetos com Restrição em Recursos (*Resource Constrained Project Scheduling Problem* – RCPSP) consiste em sequenciar todas as atividades do projeto respeitando a ordem de precedência dessas atividades, que não podem ser interrompidas. As atividades possuem tempo de processamento e consumo de recursos, que podem ser renováveis ou não renováveis. O objetivo é minimizar o *makespan*, encontrando tempos de início para cada atividade do projeto. A versão generalizada do RCPSP é o Escalonamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução (*Multi-mode Resource Constrained Project Scheduling Problem* – MRCPSP). Neste último, o objetivo permanece a minimização do *makespan*, porém as atividades podem ser processadas com várias combinações de modos de execução, o que consiste em diferentes combinações de tempo de processamento e consumo de recursos.

Em [9] é mostrado que encontrar um sequenciamento para o RCPSP é um problema NP-Difícil. Como uma generalização do RCPSP, o MRCPSP também é, portanto, um problema NP-Difícil. Além disso, os autores em [1] apontam que obter soluções viáveis para o MRCPSP é uma tarefa complicante e, de fato, os autores em [15] demonstraram que obter uma solução viável para o MRCPSP, com pelo menos 2 tipos de recursos não renováveis, é um problema NP-Completo. Conforme apontam [4], projetos com mais de 20 atividades com 3 modos de execução não podem ser solucionados na otimalidade por métodos exatos. Assim várias abordagens baseadas em heurísticas têm sido propostas para solucionar o MRCPSP.

Em [11], os autores propuseram uma implementação da metaheurística *Simulated Annealing* (SA), em que a busca local é realizada alternadamente em vizinhanças baseadas em atividades e modos de execução. Já em [12] os autores também apresentaram uma implementação de SA, tendo a busca local em dois estágios. No primeiro estágio, a vizinhança é baseada apenas em modos de execução e, para casos em que é possível melhorar o *makespan* encontrado até então, o segundo estágio é executado com vizinhanças baseadas em atividades, solucionando-se o subproblema RCPSP. Os autores em [6] usaram a estratégia evolucionária *Particle Swarm Optimization* (PSO) para atribuir modos de execução às atividades, combinando com uma busca local, que determina o sequenciamento das atividades durante a evolução do PSO. Em [13] é apresentada uma abordagem via *Differential Evolution* (DE), em que os autores restringem o tempo de busca pelo DE, comparando-o com resultados obtidos por [6] e [12]. Já os autores de [10] desenvolveram um Algoritmo Genético (AG) com modificações no cruzamento e mutação, postas a partir de observações dos mesmos em seus trabalhos com o RCPSP, assim como modificações observadas na função objetivo do AG proposto por [19]. Em [23], os autores também utilizaram um AG, combinado com a heurística *Forward/Backward Improvement* (FBI). Em [2] também é apresentado um AG, assim como uma variação da heurística FBI, mas esta última é aplicada somente em indivíduos viáveis. Em [22] é sugerida uma hibridização do AG com uma SA. Várias outras metaheurísticas têm sido propostas para solucionar o MRCPSP. Para uma revisão mais detalhada o leitor é referenciado para [14], [17], [18], [20] e [24].

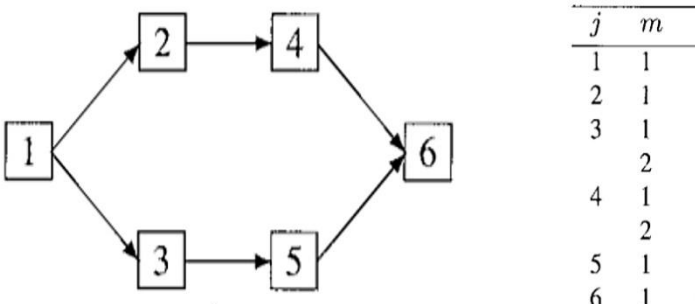
O presente artigo apresenta uma adaptação da metaheurística *Iterated Local Search* (ILS) para solucionar o MRCPSP. Como descrito em [8], este método é de simples implementação e encontra boas soluções para problemas de otimização combinatória, pois é baseado em construir sequências de ótimos locais através de (i) perturbações na solução corrente e (ii) aplicação de busca local na solução modificada pela perturbação.

O artigo está organizado da seguinte forma. O problema objeto de interesse é descrito na Seção 2. A Seção 3 apresenta a metaheurística ILS. Na Seção 4 é discutida a adaptação do ILS ao MRCPSP. A Seção 5 mostra os resultados dos experimentos computacionais realizados. A Seção 6 apresenta as conclusões derivadas dos desenvolvimentos e experimentos realizados

2. DESCRIÇÃO DO PROBLEMA

Considere um projeto com n atividades. Cada atividade j, $j = 1,...,n$ deve ser processada sem interrupção, respeitando-se a relação de precedência entre atividades, *i.e.*, cada atividade j pode ser iniciada somente após todas as suas predecessoras $p \in P_j$ estiverem finalizadas, sendo P_j o conjunto de atividades que precedem a atividade j. Convencionalmente, as atividades j_1 e j_n são fictícias, de modo que j_1 deve ser a primeira atividade a ser iniciada sem algum predecessor e j_n deve ser a última com nenhum sucessor. Ambas possuem um único modo de execução, associado com zero tempo de duração e consumo de recursos. Cada atividade deve ser executada em um modo de execução que provê o tempo de processamento e demanda de recursos para essa atividade. O número de modos em que uma atividade j pode ser executada é dado por M_j e o tempo de processamento é denotado por d_{jm}. A disponibilidade de recursos renováveis e não renováveis é dada, respectivamente, por R^{ρ}_k e R^{v}_k sendo $k = 1,...,K$ o tipo de recurso utilizado. As quantidades r^{ρ}_{jmk} e r^{v}_{jmk} definem, respectivamente, o consumo de recursos renováveis ρ e não renováveis v do tipo k no modo m pela atividade j. Considera-se que todos os parâmetros são valores inteiros não negativos. Então, o objetivo do MRCPSP é encontrar uma combinação dos modos de execução para todas as atividades, resultando em um sequenciamento com tempo de início e consumo de recursos tal que o *makespan* (duração total do projeto) seja minimizado.

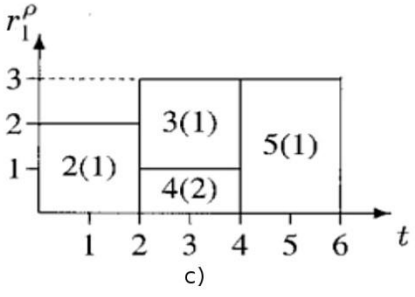
Para representar um projeto, é comum na literatura o uso da estrutura em rede *Activity on Node* (AoN) (cf. [7]). Nessa estrutura, as atividades são os nós e os arcos denotam as precedências entre atividades. A Figura 1 apresenta um exemplo de projeto nessa estrutura em rede.



a)

j	m	d_{jm}	r^{ρ}_{jm1}	K	R^{ρ}_1	R^{v}_1
1	1	0	0	{1}	3	∅
2	1	2	2			
3	1	2	2			
	2	3	1			
4	1	1	3			
	2	2	1			
5	1	2	3			
6	1	0	0			

b)



c)

Figura 1: (a) Projeto em estrutura de rede; (b) Modos de execução para atividades do Projeto; (c) Uma solução ótima. Adaptado de [3].

3. METAHEURÍSTICA *ITERATED LOCAL SEARCH* (ILS)

Dado um conjunto de soluções X, é desejado explorar X caminhando de uma solução $s \in X$ para outra que é "próxima" e "melhor" que s. De acordo com a descrição em [8], a metaheurística *Iterated Local Search* (ILS) executa esse caminho da seguinte forma. Seja s a solução corrente. Aplica-se uma perturbação em s que a leva a outra solução $s^1 \in X$. Após, uma busca local é realizada sobre s^1 conduzindo a $s^* \in X$. Se s^* passa por algum teste de aceitação, então será a nova solução corrente; caso contrário, o processo retorna para s e uma nova perturbação é executada.

A metaheurística ILS deve produzir boas amostragens de ótimos locais se as perturbações não são pequenas demais nem grande demais. Se a perturbação for pequena, o método sempre retornará para s e poucas soluções distintas serão exploradas; por outro lado, se a perturbação for grande demais, s^1 será escolhida de forma aleatória e, portanto, s^* poderá não representar boas amostragens de ótimos locais. No presente trabalho, utiliza-se o método de Descida de Primeira Melhora (*First Improvement Descent*) como técnica de busca local. Essa escolha se justifica pelo fato de que gerar um sequenciamento é o que demanda maior esforço computacional, sendo que o ILS precisa comparar as soluções encontradas, ao longo da busca, utilizando o *makespan* provido pelo sequenciamento. Então, mover-se para a primeira solução de melhora é vantajoso, devido ao custo de gerar o sequenciamento. Na próxima seção detalha-se como esse método percorre a vizinhança.

4. ILS APLICADO AO MRCPSP

Esta seção apresenta o detalhamento da adaptação do ILS para solucionar o MRCPSP.

4.1. REPRESENTAÇÃO DAS SOLUÇÕES

A representação computacional de uma solução é um componente importante para o ILS. De acordo com [8], a representação deve ser tal que permita transições entre soluções vizinhas, rápida validação de soluções e, além disso, deve garantir acesso ao espaço de soluções.

Várias estruturas de representação de soluções para o MRCPSP podem ser encontradas na literatura, como as apresentadas em [7] e em [17]. No presente artigo, uma solução s é representada por um par de vetores $s = (J, M)$, em que J é uma permutação viável de todas as atividades e M uma permutação válida de modos de execução. Assim, os elementos $j_i \in J$ e $m_i \in M$ informam que, na posição i, a atividade j é executada pelo modo m. A Figura 2 mostra esse esquema de representação. Já a Figura 3 apresenta um exemplo de solução para o Problema de Projeto apresentado na Figura 1.

Atividade	j_1	j_2	-	-	-	J_n
Modo	m_1	m_2	-	-	-	M_n

Figura 2: Esquema de representação de solução

Atividade	1	2	4	3	5	6
Modo	1	1	2	1	1	1

Figura 3: Exemplo de solução para o Problema de Projeto representado na Figura 1.

4.2. FUNÇÃO DE AVALIAÇÃO

A função objetivo é frequentemente utilizada como função de avaliação, pois provê uma medida de qualidade para uma dada solução. Quando o espaço de soluções inviáveis é considerado, a função de avaliação utiliza a função objetivo com o acréscimo de uma penalidade.

Assim sendo, a função de avaliação foi discutida na utilização de heurísticas aplicadas ao MRCPSP com minimização do *makespan*. Em [19], definiu-se uma função de avaliação considerando o *makespan* para soluções viáveis. Para as inviáveis, foi utilizado um *upper bound* conhecido como horizonte do projeto, com adição da quantidade de recursos não renováveis consumida em excesso. Essa mesma função de avaliação foi usada em [11]. Entretanto, os autores de [10] apontaram que, usando essa avaliação, soluções inviáveis com *makespan* distintos e consumindo a mesma quantia em excesso de recursos não renováveis irão receber a mesma penalização. Em consequência, propuseram uma nova função de avaliação para soluções inviáveis, considerando o excesso de consumo de recursos não renováveis, um *lower bound*, conhecido como caminho crítico, o *makespan* da solução avaliada e o máximo *makespan* das soluções na população. Os autores de [23] utilizaram essas duas funções de avaliação. Entretanto, em [2] observou-se que essa última avaliação adiciona unidades de tempo do *makespan* a unidades de recursos a partir do excesso consumido de consumo de recursos não renováveis. Claramente, a magnitude de ambos os aspectos podem distorcer o significado da solução. Assim, os autores em [2] apresentaram uma função de avaliação que mantém a mesma magnitude entre as parcelas envolvidas na função. A mesma função de avaliação é adotada em [22]. Portanto, no presente artigo, essa última função de avaliação é utilizada, no entanto, com a troca do máximo *makespan* pelo horizonte do projeto. Então, o excesso consumido de recursos não renováveis por uma solução *s*, dado por *RC(s)*, e a função de avaliação adotada, dada por *f(s)*, são definidos como:

$$RC(s) \leftarrow \sum_{\forall k \in R_k^{\nu}} max\left\{0, \sum_{j=1}^{n} r_{jmk} - R_k^{\nu}\right\} \tag{1}$$

$$f(s) = \begin{cases} 1 - \dfrac{UB - mak(s)}{UB}, & se\ RC(s) = 0 \\ 1 + \dfrac{mak(s) - LB}{mak(s)} + \displaystyle\sum_{\forall k \in R_k^{\nu}} max\left\{0, \dfrac{\sum_{j=1}^{n} r_{jmk} - R_k^{\nu}}{R_k^{\nu}}\right\}, & caso\ constrário \end{cases} \tag{2}$$

Na Expressão (2), UB é o horizonte do projeto e LB é o caminho crítico.

4.3. ESTRUTURAS DE VIZINHANÇA

Existem dois tipos de vizinhanças para o MRCPSP: (i) vizinhança baseada em atividades e (ii) vizinhança baseada em modos de execução. Nesse trabalho, utilizam-se dois movimentos para explorar o espaço de soluções através dessas duas estruturas de vizinhança. Estes dois movimentos são descritos a seguir.

4.3.1. Mudança de Um Modo de Execução - MUME

Dado um vetor de modos de execução, esse movimento troca o modo de uma atividade quando possível. Considerando o exemplo de Problema de Projeto apresentado na Figura 1, a Figura 4 apresenta uma solução *s* e um de seus vizinhos *s'* ao aplicar o movimento MUME sobre *s*.

s

Atividade	1	2	4	3	5	6
Modo	1	1	2	1	1	1

s'

Atividade	1	2	4	3	5	6
Modo	1	1	2	2	1	1

Figura 4: Movimento MUME aplicado a *s* levando a um vizinho *s'*

4.3.1.1. *Shift* em uma Atividade - SUA

Ao trocar uma atividade de posição, é necessário garantir que a relação de precedência desta atividade, para todos seus predecessores e sucessores, seja respeitada. Então, sua posição pode ser alterada para trás ou para frente, desde que a relação de precedência seja garantida. Para uma atividade *j*, seu predecessor e sucessor mais próximos, definidos, respectivamente, como $\bar{j}_1 \in P_j$ e $\bar{j}_2 \in S_j$, são encontrados determinando-se o quão longe *j* pode mover-se para frente ou para trás. Assim, a atividade *j* pode movimentar-se para trás até que $\bar{j}_1$ seja alcançado, ou para frente, até que $\bar{j}_2$ seja alcançado. A Figura 5 apresenta um exemplo desse movimento, considerando a situação posta pelo Problema de Projeto dado pela Figura 1.

s

Atividade	1	2	4	3	5	6
Modo	1	1	2	1	1	1

s'

Atividade	1	2	3	5	4	6
Modo	1	1	1	1	2	1

Figura 5: Movimento SUA aplicado a *s* levando a um vizinhos *s'*.

4.4. PERTURBAÇÕES

A literatura associada ao MRCPSP mostra que movimentos aplicados em modos de execução são mais sensíveis para explorar outras regiões do espaço de busca. Assim, propõe-se, no presente artigo, uma classificação para perturbações, que possui cinco níveis:

- Nível 1: Executar MUME;
- Nível 2: Executar MUME para duas atividades distintas;
- Nível 3: Executar MUME para três atividades distintas;
- Nível 4: Executar MUME para quatro atividades em sequência na representação AoN;
- Nível 5: Executar MUME para cinco atividades em sequência na representação AoN;

Após a perturbação ser executada em um nível, a busca local faz a intensificação naquela região. Se a busca local não for capaz de encontrar melhorias no nível atual após *k* tentativas, então o próximo nível é escolhido para novas tentativas em melhorar a solução corrente.

4.5. O ALGORITMO ILS-MRCPSP

Uma solução inicial é obtida através da regra de prioridade aleatória. Regras de prioridade definem uma ordem em que as atividades podem ser selecionadas para o sequenciamento, de forma a respeitar a ordem de precedência. Para a regra de prioridade aleatória, uma atividade é selecionada, aleatoriamente, para o sequenciamento, somente se todos seus predecessores já estiverem sequenciados. Uma revisão a respeito de regras de prioridade é apresentada por [21]. Em seguida, o mesmo critério é utilizado para determinar os modos de execução, *i.e.,* modos aleatórios são determinados para cada atividade. Definido o modo de execução para cada atividade, assim como a ordem em que as atividades serão selecionadas, o esquema de geração de *schedules* (*Schedule Generation Scheme* – SGS) em série (cf. [7]) constrói o sequenciamento inicial. O SGS utilizado neste trabalho é o mesmo utilizado pelos autores em [5]. Então, a busca local explora a vizinhança da solução atual. Finalmente, itera-se entre perturbações e intensificações através do ILS. O método de Descida de Primeira Melhora realiza a busca local, utilizando os movimentos MUME e SUA; contudo, tal busca é realizada em dois estágios. Primeiro, é explorado a vizinhança usando somente o movimento MUME. Com os modos fixados, resolve-se, então, o subproblema RCPSP, através do movimento SUA. Os critérios de parada utilizados são baseados em uma quantidade fixa de tempo e o melhor valor conhecido pela literatura. O primeiro, critério de parada baseado em tempo, é dado como *(M|J|)* segundos, em que *M* é uma constante definida para cada classe de instâncias testadas e |*J*| é a quantidade de atividades presente nas instâncias. Já o critério de parada dado pela determinação da melhor solução conhecida é satisfeito assim que esse valor conhecido é alcançado pela metaheurística ILS-MRCPSP. Há, também, a quantidade de tentativas em melhorar a solução corrente para cada nível de perturbação. Essa quantia foi definida empiricamente como *5%(M|J|)*. O algoritmo ILS-MRCPSP é apresentado abaixo.

```
Result: s*
Gerar sequenciamento inicial s ;
s* ↩ realizar busca local em dois estágios em s ;
while critério de parada não é satisfeito do
    for k ← 1 to MAX_LEVEL do
        s^1 ↩ Aplicar perturbação em s* para nível k ;
        repeat
            Aplicar a busca local em dois estágios em s^1 ;
            if f(s^1) < f(s*) then
                s* ↩ s^1 ;
                reiniciar k ;
            end
        until número máximo de tentativas foi alcançado OU s* foi melhorada ;
    end
end
```

Figura 6: Algoritmo ILS-MRCPSP.

4.6. PRÉ-PROCESSAMENTO

Antes de iniciar o ILS-MRCPSP, realiza-se o procedimento de redução das soluções, descrito em [3], para remover modos ineficientes ou não executáveis e recursos redundantes. De acordo com [3], um modo é não executável se sua execução pode violar as restrições dos recursos renováveis e não renováveis em qualquer sequenciamento. Um modo é ineficiente se existe outro modo, para a mesma atividade, com o menor (ou mesmo) tempo de duração e menor (ou mesmo) consumo de recursos de todos os tipos. Um recurso não renovável é redundante se a soma das máximas demandas de cada atividade para esse recurso não excede sua disponibilidade. A exclusão desses modos ineficientes e não executáveis e dos recursos não renováveis redundantes, segundo [3], não afeta o conjunto de soluções viáveis ou *schedules* ótimos. Como esse procedimento apresenta baixo custo computacional, o mesmo foi utilizado como parte do algoritmo proposto.

5. RESULTADOS COMPUTACIONAIS

O ILS-MRCPSP foi implementado em linguagem C, usando compilador GNU GCC 4.8.2. Os testes foram realizados em um computador com processador Intel Core i3-3217U; 1.80GHz; 4GB RAM; e sistema operacional Ubuntu Linux 64bits.

Foram utilizadas as classes de instâncias da biblioteca pública *Project Scheduling Problem Library (*PSPLIB) (cf. [16]). O algoritmo foi testado sobre os conjuntos J10, J20 e J30. Para as classes J10 e J20, o tempo de processamento foi fixado em $2|J|$ segundos. Valores ótimos são conhecidos para esses grupos de instâncias. Para o conjunto J30, foi fixado o tempo de processamento em $3.5|J|$ segundos. Soluções para esse conjunto de instâncias foram encontradas, apenas, pela utilização de métodos heurísticos. As características dessas classes são apresentadas na Tabela 1.

Classe	**# de instâncias viáveis**	**# de atividades**	**# de modos de execução**	**# máxima de sucessores por atividade**	**# de recursos renováveis**	**# de recursos não renováveis**
J10	536	10	3	3	2	2
J20	554	20	3	3	2	2
J30	552	30	3	3	2	2

Tabela 1: Características das classes de instâncias PSPLIB utilizadas.

Os resultados computacionais alcançados pela aplicação do algoritmo ILS-MRCPSP são apresentados na Tabela 2. Nesta Tabela, o desvio para as melhores soluções conhecidas na literatura, ou seja, soluções alcançadas por métodos exatos ou por técnicas heurísticas, é calculado como:

$$Desv_{medio} = \frac{Media - Melhor_{lit}}{Melhor_{lit}} \times 100 \quad (3)$$

Na Expressão (3), $Desv_{medio}$ indica o percentual do desvio do valor médio das soluções, para as 30 execuções, em relação ao melhor valor conhecido na literatura.

Classe	# instâncias testadas	ILS-MRCPSP			
		Desvio médio (%)	Determinação das melhores soluções conhecidas (%)	Soluções viáveis (%)	Tempo médio (s)
J10	100	0.820	88.800	100	2.230
J20	100	1.149	76.933	100	9.996
J30	80	2.126	63.790	100	-

Tabela 2: Resultados nas 30 execuções do ILS-MRCPSP para problemas das classes J10, J20 e J30

Para os conjuntos J10 e J20, foram testadas 100 instâncias. Para o conjunto J30, foram testadas 80 instâncias. Pela Tabela 2, em relação às instâncias testadas, dado que a solução inicial é aleatória e existe a possibilidade da busca iniciar com uma solução inviável, o algoritmo ILS-MRCPSP apresentou convergência em tempo aceitável para os grupos testados, sendo que 88% e 76% das soluções ótimas foram alcançadas nas 30 execuções, respectivamente, para as classes J10 e J20, com aproximadamente 1% de desvio médio para os valores ótimos. Isso indica que, se o algoritmo ILS-MRCPSP não finalizava pelo critério de encontrar a solução ótima, a solução final encontrada por esse algoritmo, dado pelo critério de parada fixado pelo tempo, foi de boa qualidade. Para a classe J30, não são apresentados os valores de tempo médio, pois, como posto, os valores conhecidos para esse grupo foram obtidos pela utilização de técnicas heurísticas e não estão relacionados, portanto, a valores ótimos presentes na literatura. Não foram determinadas novas soluções para este conjunto de instâncias. Contudo, foram encontradas 63% das melhores soluções conhecidas, com desvio médio de 2,12% para esses valores conhecidos da literatura, com tempo total de execução de $3.5|J| = 105$ segundos. Conforme observado pela literatura, os resultados apresentados pela Tabela 2 podem apresentar melhorias, para as classes testadas, se for aceito maior tempo de execução.

Na literatura, não há informação sobre como as instâncias estão agrupadas em todas as instâncias avaliadas. Então, o fato do algoritmo ILS-MRCPSP terminar sua execução com 100% de soluções viáveis e com boa qualidade, o torna um método eficiente, uma vez que permite iniciar a busca com soluções inviáveis ou de baixa qualidade. Os trabalhos citados, com aplicação de metaheurísticas, utilizam de procedimentos heurísticos para viabilizar as soluções inviáveis durante a busca, considerando apenas o espaço de soluções viáveis, ou viabilizam somente a solução inicial, porém considerando todo o espaço de soluções. Considerar somente o espaço de soluções viáveis, mesmo que as instâncias da PSPLIB favoreçam viabilizar soluções inviáveis, devido à baixa quantidade de modos de execução, é uma tarefa computacionalmente difícil, *i.e.*, viabilizar soluções inviáveis ou procurar por essas soluções é um problema NP-Completo. Essa foi a razão pela qual, no presente artigo, nenhuma solução inviável foi viabilizada, exceto pela busca local.

6. CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho tratou o problema de escalonamento em projetos com restrição em recursos e múltiplos modos de execução, com o objetivo de minimizar o *makespan*. Dado a complexidade do problema, foi proposto a metaheurística *Iterated Local Search* (ILS) para solucioná-lo. Na abordagem proposta neste artigo, uma solução inicial é gerada de forma aleatória e então refinada pela busca local através do método de Descida de Primeira Melhora. Para exploração do espaço de soluções, são utilizados níveis de perturbações para diversificação da solução corrente, em que, em seguida, essa solução é refinada pela busca local. O passo de diversificação e intensificação é iterativo na busca do ILS. Porém, diferentemente do convencional, a busca local foi realizada em dois estágios, explorando

vizinhanças em modos de execução e, em seguida, exploração por vizinhanças baseadas em atividades.

Três classes de instâncias da PSPLIB foram submetidas a testes. Os resultados mostram que o algoritmo proposto foi capaz de determinar soluções conhecidas, de um grupo de instâncias testadas, em curto tempo de processamento. Quando a melhor solução conhecida pela literatura não foi encontrada, o algoritmo proposto foi capaz de determinar boas soluções, dado pela qualidade dessas soluções evidenciada pelo desvio médio para o melhor valor conhecido.

Como propostas futuras, propõe-se o estudo de heurísticas para a inicialização do método com soluções viáveis, mesmo considerando todo o espaço de busca. Adicionalmente, outras metaheurísticas podem ser exploradas, assim como, a paralelização dessas técnicas.

AGRADECIMENTOS

Os autores agradecem às agências CAPES, FAPEMIG e CNPq, assim como ao CEFET-MG, pelo apoio ao desenvolvimento do trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Andreas Drexl e Juergen Gruenewald. *Nonpreemptive multi-mode resource-constrained project scheduling*. IIE transactions, 25(5): 74–81, 1993.

[2] Antonio Lova, Pilar Tormos, Mariamar Cervantes, e Federico Barber. *An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes*. International Journal of Production Economics, 117(2):302–316, 2009.

[3] Arno Sprecher, Sonke Hartmann, e Andreas Drexl. *An exact algorithm for project scheduling with multiple modes*. Operations-Research-Spektrum, 19(3):195–203, 1997.

[4] Arno Sprecher e Andreas Drexl. *Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm*. European Journal of Operational Research, 107(2):431–450, 1998.

[5] Asta Shahriar, Daniel Karapetyan, Ahmed Kheiri, Ender Özcan, e A. Parkes. *Combining monte-carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem.* In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, 836-839, 2013.

[6] Bassem Jarboui, Najeh Damak, Patrick Siarry, e A Rebai. A *combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems*. Applied Mathematics and Computation, 195(1):299–308, 2008.

[7] Erik L Demeulemeester. *Project scheduling: a research handbook*, volume 102. Springer, 2002.

[8] Helena Lourenço, Martin Olivier, e Thomas Stützle. *Iterated local search*. In "*Handbook of metaheuristics", Ed.* Fred Glover e Gary A Kochenberger. Springer Science & Business Media, 2003.

[9] Jacek Blazewicz. *Scheduling under resource constraints: Deterministic models*, volume 7. JC Baltzer, 1986.

[10] Javier Alcaraz, Concepcion Maroto, e Ruben Ruiz. *Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms*. Journal of the Operational Research Society, 54(6):614–626, 2003.

[11] Joanna Jozefowska, Marek Mika, Rafal Rózycki, Grzegorz Waligóra, e Jan Weglarz. *Simulated annealing for multi-mode resource-constrained project scheduling*. Annals of Operations Research, 102(1-4):137–155, 2001.

[12] Klein Bouleimen e Housni Lecocq. *A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version*. European Journal of Operational Research, 149(2):268–281, 2003.

[13] Najeh Damak, Bassem Jarboui, Patrick Siarry, e Taicir Loukil. *Differential evolution for solving multi-mode resource-constrained project scheduling problems*. Computers & Operations Research, 36(9):2653–2659, 2009.

[14] Peter Brucker, Andreas Drexl, Rolf Mohring, Klaus Neumann, e Erwin Pesch. *Resource-constrained project scheduling: Notation, classification, models, and methods.* European journal of operational research, 112(1):3–41, 1999.

[15] Rainer Kolisch e Andreas Drexl. *Local search for nonpreemptive multi-mode resourceconstrained project scheduling*. IIE transactions, 29(11): 987–999, 1997.

[16] Rainer Kolisch e Arno Sprecher. *Psplib - a project scheduling problem library*. European Journal of Operational Research, 96(1):205–216, 1997.

[17] Rainer Kolisch e Sonke Hartmann. *Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis*. Springer, 1999.

[18] Rainer Kolisch e Sonke Hartmann. *Experimental investigation of heuristics for resourceconstrained project scheduling: An update*. European Journal of Operational Research, 174(1):23–37, 2006.

[19] Sonke Hartmann. *Project scheduling with multiple modes: a genetic algorithm*. Annals of Operations Research, 102(1-4):111–135, 2001.

[20] Sonke Hartmann e Rainer Kolisch. *Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem*. European Journal of Operational Research, 127(2):394–407, 2000.

[21] Tyson R Browning e Ali A Yassine. *Resource-constrained multi-project scheduling: Priority rule performance revisited*. International Journal of Production Economics, 126(2):212–228, 2010.

[22] Vijay S Bilolikar, Karuna Jain, e Mahesha Sharma. *An annealed genetic algorithm for multi mode resource constrained project scheduling problem*. International Journal of Computers and Applications, 60(1):36–42, 2012.

[23] Vincent Van Peteghem e Mario Vanhoucke. *A genetic algorithm for the preemptive and nonpreemptive multi-mode resource-constrained project scheduling problem*. European Journal of Operational Research, 201(2):409–418, 2010.

[24] Vincent Van Peteghem e Mario Vanhoucke. *An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances*. European Journal of Operational Research, 235(1):62–72, 2014.