Multithreading Iterated Local Search aplicado ao Problema de Horários Escolares

Landir Saviniec[⋆], Maristela Oliveira Santos[⋆], Alysson Machado Costa[†], Ademir Aparecido Constantino[‡]

landir.saviniec@gmail.com, mari@icmc.usp.br,
alysson.costa@unimelb.edu.au, ademir@din.uem.br

[⋆]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo – Brasil
[†]School of Mathematics and Statistics – The University of Melbourne – Australia
[‡]Departamento de Informática – Universidade Estadual de Maringá – Brasil

RESUMO

Este artigo aborda o Problema de Horários Professor-Turma (PPT). Trata-se de uma variante da classe de problemas conhecida na literatura por *High School Timetabling Problem*. O problema consiste em alocar um conjunto de aulas professor-turma em períodos durante a semana. O problema é NP-Difícil e devido a diversos requisitos práticos, casos reais geralmente apresentam um número elevado de variáveis, tornando ineficiente a aplicabilidade de métodos exatos. Deste modo, métodos heurísticos são comumente empregados. O desenvolvimento de boas heurísticas deve levar em consideração o balanceamento entre dois fatores importantes: estratégias de intensificação e diversificação. No estudo apresentado neste artigo, investigamos o uso de *multithreading* no método *Iterated Local Search* com objetivo de intensificar a busca em torno da melhor solução e empregamos lista tabu para minimizar problemas de ciclagem durante descida em busca local. Conduzimos vários experimentos computacionais com instâncias reais do problema. Os resultados mostraram uma melhora significativa tanto em qualidade de soluções como em tempo computacional em relação a versão desprovida das estratégias propostas.

PALAVRAS CHAVE. Problema de Horários Professor-Turma, *Iterated Local Search*, *Multithreading*.

Área Principal: PO na Educação, Metaheurísticas, Otimização Combinatória.

ABSTRACT

This paper addresses the Class-Teacher Timetabling Problem, which is a variant of the High School Timetabling Problem. The problem consists in scheduling a set of class-teacher lessons in periods during the week. The problem is NP-Hard and due to various practical requirements, real cases generally have a large number of variables, making ineffective the applicability of exact methods. Thus, heuristic methods are commonly employed in solving the problem. The development of good heuristics should take into account the trade-off between two important issues: intensification and diversification of the search. In the study presented here, we investigated the use of multithreading into Iterated Local Search to intensify the search around the incumbent best solution and the use of tabu list to minimize cycles during local search descents. We have conducted a series of computational experiments with real instances of the problem. The results showed significant improvements both on the quality of solutions and on computational times when compared to a previous version of the method that do not employ such strategies.

KEYWORDS. Class-Teacher Timetabling Problem, Iterated Local Search, Multithreading.

Main Area: OR in Education, Metaheuristics, Combinatorial Optimization.

1. Introdução

O Problema de Horários Professor-Turma (PPT) é uma variante da classe de problemas de horários em ambientes educacionais. Na literatura essa classe de problemas é conhecida por *timetabling* e está dividida em três sub-áreas: Problema de Horários de Cursos Universitários (LEWIS; PAECHTER; MCCOLLUM, 2007) (GASPERO; SCHAERF; MCCOLLUM, 2007), Problema de Horários de Exames (MCCOLLUM et al., 2007) (QU et al., 2009) e Problema de Horários Escolares (SCHAERF, 1999b) (POST et al., 2012) (PILLAY, 2013). O PPT pertence a esta última sub-área.

O Problema de Horários Escolares (PHE) é um tipo de problema encontrado em escolas de ensino fundamental e médio enquanto as outras duas categorias são mais comuns em ambientes universitários. O PHE possui inúmeras variantes. Em algumas, as turmas possuem salas dedicadas e não há necessidade de alocação das mesmas (SANTOS; OCHI; SOUZA, 2005) (SOUSA; MORETTI; PODESTÁ, 2008). Em outras, as salas disponíveis não são dedicadas e há alta taxa de ocupação, havendo portanto, a necessidade de uma alocação eficiente de salas (JACOBSEN; BORTFELDT; GEHRING, 2006) (WILKE; OSTLER, 2008) (POST; AHMADI; GEERTSEMA, 2010). O problema também se divide em outras duas variantes em relação a forma como os professores são escolhidos para lecionar as disciplinas de cada turma. Na maioria dos casos encontrados na literatura, essa decisão é tomada antes do processos de construção da grade de horários (SCHAERF, 1999a) (SANTOS; OCHI; SOUZA, 2005) (SOUSA; MORETTI; PODESTÁ, 2008). Por outro lado, há casos em que a decisão é tomada durante o processo de alocação (ALVAREZ-VALDÉS; PARREÑO; TAMARIT, 2002).

O PPT é caracterizado pela existência de salas dedicadas e associação prévia dos professores as disciplinas das turmas. Nas últimas duas décadas, vários estudos de caso abordando o PPT em escolas brasileiras surgiram na literatura (SOUZA; COSTA; GUIMARÃES, 2002) (SOUZA; OCHI; MACULAN, 2003) (MOURA et al., 2004) (COELHO; SOUZA, 2006) (SOUSA; MORETTI; PODESTÁ, 2008). Isto mostra que o estudo desta variante é de importância no sistema educacional brasileiro.

Em situações reais, diversos requisitos são necessários para atender preferências pedagógicas ou do corpo docente. Isto aumenta drasticamente o número de variáveis e torna a aplicabilidade de métodos exatos parcial ou totalmente ineficientes para resolver o problema. Deste modo, uma grande quantidade de estudos tem desenvolvido heurísticas para tratar o problema (PILLAY, 2013). Uma categoria de heurísticas bastante empregada é baseada em vizinhanças. Algumas questões bastante exploradas no desenvolvimento desses métodos, mas que introduzem novas dificuldades a cada nova aplicação, são: o uso de estratégias de intensificação, diversificação e mecanismos para evitar ciclagem.

No campo do PHE, *Tabu Search* (SOUZA; OCHI; MACULAN, 2003) (SANTOS; OCHI; SOUZA, 2005) (JACOBSEN; BORTFELDT; GEHRING, 2006) (BELLO; RANGEL; BOERES, 2008), *Iterated Local Search* (SAVINIEC et al., 2013) (FONSECA et al., 2014), *Variable Neighborhood Search* (KOCHETOV; KONONOVA; PASCHENKO, 2008) (BRITO et al., 2012) (FONSECA; SANTOS, 2014) e *Simulated Annealing* (WILKE; OSTLER, 2008) (ZHANG et al., 2010) (BRITO et al., 2012) tem sido as meta-heurísticas mais empregadas.

Em (SAVINIEC et al., 2013), foi proposto um método baseado em *Iterated Local Search* (ILS) para tratar problemas com características do PPT. O método foi testado com instâncias do problema proposto por (SOUZA; OCHI; MACULAN, 2003) e os resultados foram comparados com limitantes duais obtidos pelo método de Geração de Cortes e Colunas proposto por (SANTOS et al., 2012). A principal contribuição do trabalho foi encontrar soluções primais que ajudaram provar a otimalidade de três instâncias em aberto. Além disso, o método encontrou soluções ótimas em todas as instâncias consideradas. O trabalho realizado em (SAVINIEC et al., 2013) mostrou que ILS é uma potencial ferramenta para tratar problemas do tipo PPT. No estudo apresentado neste artigo, estendemos esta técnica através do uso de lista tabu para minimizar ciclagem durante

descida em busca local e de *multithreading* para intensificar a busca em torno da melhor solução. A proposta é comparada com a abordagem anterior usando várias instâncias reais de um PPT diferente da versão de (SOUZA; OCHI; MACULAN, 2003).

2. O Problema de Horários Professor-Turma (PPT)

O PPT considerado aqui é baseado em casos reais de escolas públicas brasileiras. Nessas escolas as atividades escolares são distribuídas sobre um conjunto D de dias durante a semana (segunda a sexta-feira). Cada dia de atividade é dividido em três turnos (matutino, vespertino e noturno), cada um contendo um conjunto H de períodos de aula (5 períodos). Um quadro de horários é produzido para cada turno de forma independente. Em cada turno existe um conjunto T de turmas e um conjunto P de professores. Turmas são grupos disjuntos de estudantes matriculados em um mesmo conjunto de disciplinas (ex.: português, matemática, química e etc.). Cada disciplina de uma turma possui uma carga horária semanal e é lecionada por um único professor. A escolha dos professores para lecionar as disciplinas é feita previamente pela escola. Turmas estão integralmente ocupadas em todos os períodos de aula durante a semana e possuem salas dedicadas. Professores não possuem dedicação exclusiva e em geral trabalham em outras escolas. Deste modo, é desejável que eles possuam uma agenda semanal compacta e sejam alocados somente em períodos que estão disponíveis para lecionar. Além disso, a escola almeja uma compactação balanceada, de forma a evitar professores com agenda mais compacta que outros. Logo, o problema consiste em alocar as aulas das disciplinas das turmas em $|D| \times |H|$ períodos de aula durante a semana.

Os dados que formam uma instância do PPT são definidos por:

Definição 1 (Instância do PPT) *Uma instância do PPT se refere aos dados de entrada para a programação dos horários das aulas de um determinado turno, sendo representada por dois conjuntos:*

- *Um conjunto* $R = \{\langle t, p, \theta, \delta, \pi\rangle | t \in T,\ p \in P,\ \theta \in \mathbb{N},\ \delta \in \mathbb{N}\ e\ \pi \in \mathbb{N}\}$*, denominado conjunto de requerimento de aulas. Um requerimento* $r \in R$ *(*$r = \langle t_r, p_r, \theta_r, \delta_r, \pi_r\rangle$*) indica que a turma* t_r *requer* θ_r *aulas com o professor* p_r*,* δ_r *é o número máximo de aulas diárias e* π_r *é o número mínimo de aulas duplas consecutivas requisitadas no requerimento* r*.*
- *Um conjunto* $U = \{\langle p, d, h\rangle | p \in P,\ d \in D,\ h \in H_d\}$*, denominado conjunto de períodos de indisponibilidade dos professores. Uma tripla* $\langle p, d, h\rangle$ *indica que o professor* p *está indisponível no período* h *do dia* d*.*

Dado uma instância do PPT, o problema consiste em produzir um quadro de horários Z para as aulas requeridas em R. Dois tipos de restrições são consideradas: fortes e fracas.

Definição 2 (Restrições fortes) *São aquelas que devem ser satisfeitas para que uma solução seja factível, ou seja, passível de uso pela escola. O conjunto de restrições fortes é definido por:*

c_1 : *Todo requerimento* $r \in R$ *deve ter* θ_r *aulas alocadas;*
c_2 : *Uma turma* $t \in T$ *deve assistir uma aula por período;*
c_3 : *Um professor* $p \in P$ *deve lecionar no máximo uma aula por período;*
c_4 : *Um professor* $p \in P$ *não deve ser alocado para lecionar em períodos nos quais esteja indisponível.*

Definição 3 (Restrições fracas) *São aquelas cujo número de violações deve ser minimizado para que o quadro de horários apresente qualidade. Este conjunto é definido por:*

c_5 : *Todo requerimento* $r \in R$ *deve ter no máximo* δ_r *aulas alocadas por dia;*
c_6 : *Aulas não consecutivas de um requerimento* $r \in R$ *em um mesmo dia devem ser*

evitadas;
c_7 : *Um número mínimo de aulas duplas consecutivas π_r requisitadas em um requerimento $r \in R$ deve ser atendido sempre que possível;*
c_8 : *Professores não devem ter janelas*[1] *em sua agenda diária. Períodos para os quais o professor está indisponível não são considerados janelas;*
c_9 : *A carga horária semanal do professor deve ser concentrada em um número mínimo de dias (compactação);*
c_{10} : *A compactação na agenda semanal dos professores deve ser balanceada, de forma a evitar que alguns professores tenham uma agenda mais compacta que outros (balanceamento).*

3. O método *Iterated Local Search*

Iterated Local Search é um método de busca local baseado em vizinhanças e foi proposto por (LOURENÇO; MARTIN; STÜTZLE, 2003). O método consiste em aplicar, a cada iteração, uma busca local para encontrar um mínimo local, seguida de uma perturbação para escapar deste mínimo.

Em (SAVINIEC et al., 2013) a versão proposta (Algoritmo 1) emprega vizinhanças geradas por um operador de vizinhança baseado em *Kempe chain interchanges*[2] (Figura 1). A busca local é baseada na técnica *First Improvement* (linha 7). Isto é, ao verificar a vizinhança durante a descida, o primeiro vizinho melhor ou igual a solução corrente é aceito. Essa estratégia de busca local gera ciclos após a perturbação e ciclos durante a descida.

Algoritmo 1 Método ILS adaptado de (SAVINIEC et al., 2013)

ILS-TQ$(Z_0, Iter)$

```
1  Z = LocalSearch(Z_0)
2  Z* = Z                                   // melhor solução
3  NotImproved = 0
4  i = 0
5  while (i < Iter) do
6      Z = Perturbation(Z, 1)
7      Z = LocalSearch(Z)
8      if f(Z) < f(Z*) then
9          NotImproved = 0
10     else
11         NotImproved = NotImproved + 1
12     if f(Z) ≤ f(Z*) then                 // critério de aceitação
13         Z* = Z
14     if NotImproved ≥ 3 then              // se não houve melhoras após três iterações
15         Z = Z*                           // retornar para Z*
16         NotImproved = 0
17     i = i + 1
18 return Z*
```

No primeiro caso, após perturbação na linha 6, o primeiro vizinho aceito na descida é geralmente, a solução anterior a perturbação. Isto se deve ao uso de um único movimento de perturbação. Neste ponto é interessante destacar que perturbações maiores foram testadas e não apresentaram bons resultados, pois causam efeitos de "*re-starts*" na solução. Isto acontece porque o operador de vizinhança introduz, na maioria das vezes, uma série de movimentos necessários para corrigir infactibilidades na solução. No segundo caso, ciclos são causados pela aceitação de soluções planas. Note que durante a descida, soluções melhores ou iguais são aceitas.

[1] Períodos ociosos entre dois períodos de atividade em um dado dia.
[2] Cadeia de movimentos.

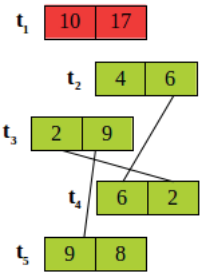
Para atenuar a limitação causada por ciclos, introduzimos uma lista tabu para memorizar os *movimentos de perturbação* e os *movimentos de descida*. No início de cada iteração, a lista é apagada antes da linha 6 e os movimentos de perturbação são inseridos na lista. Durante a descida em busca local, na linha 7, os movimentos são aceitos se eles não são tabu ou melhoram a solução Z^*. Movimentos aceitos também se tornam tabu.

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	5	10	15	17	15
t_2	1	4	5	6	7
t_3	9	2	1	9	6
t_4	10	6	12	2	11
t_5	6	9	9	8	2

(a) Solução corrente Z. Nesta figura as linhas representam os professores que lecionam as turmas em cada período de aula.

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	5	10	15	17	15
t_2	1	6	5	4	7
t_3	9	9	1	2	6
t_4	10	2	12	6	11
t_5	6	8	9	9	2

(b) Exemplo de solução vizinha obtida permutando-se os professores da componente conexa identificada com a cor verde no grafo de conflitos.



(c) Grafo de conflitos. Este grafo permite realocar professores entre dois períodos distintos sem violar a restrição c_3. Note que cada componente conexa do grafo gera um vizinho da solução Z.

Figura 1: Exemplo do operador TQ empregado em (SAVINIEC et al., 2013).

Em nossa proposta um movimento tabu é dado por uma n-upla $\langle slot_i, slot_j, f, index \rangle$, com $i, j = 1, ..., |D| \times |H|$ e $i \neq j$. Onde um $slot_i$ é um período de aula $\langle d, h \rangle$, $d \in D$ e $h \in H$. A componente f é o valor de função objetivo e $index$ é um índice que representa uma componente conexa no grafo de conflitos.

Com objetivo de intensificar a busca em torno da melhor solução Z^* em cada iteração, introduzimos o uso de busca local *multithreading*. Isto é, em vez de executar uma única busca local na linha 7, múltiplas *threads* são executadas. O método é composto por uma *thread* mãe (Algoritmo 2) e *threads* filhas (Algoritmo 3). Após cada *thread* filha retornar um mínimo local na linha 12 da *thread* mãe, a melhor solução Z é selecionada na linha 13. Se Z for melhor ou igual a Z^*, ela passa a ser a melhor solução atual. Após três iterações sem melhoras, toda *thread* k tem sua solução $solVector[k]$ reinicializada a partir de Z^* pela *thread* mãe, linha 22.

3.1. Representação de soluções e função objetivo

Uma solução do PPT é representada por uma matriz tridimensional $Z_{|T|\times|D|\times|H|}$ de valores inteiros não negativos, onde uma célula $z_{tdh} \in \{1, 2, \ldots, |P|\}$ indica o professor designado para lecionar a turma $t \in T$ no período $h \in H$ do dia $d \in D$. Esta estrutura de dados trata automaticamente as restrições c_1 e c_2.

A função objetivo (Expressão 1) equivale a soma ponderada do número de violações em cada restrição do problema:

$$min \quad f(Z) = \sum_{i=3}^{10} \alpha_i \cdot \beta_i \tag{1}$$

Na Expressão 1, a constante α_i é a penalidade aplicada ao número de violações β_i ocorrido na i-ésima restrição. Os termos para $(i = 3, 4)$ medem o nível de factibilidade e os termos para $(i = 5, ..., 10)$ medem a qualidade da solução.

Algoritmo 2 Método ILS – *thread* mãe

ILS-MAIN($Z_0, Iter, Nthreads$)

```
1  Inicializar thVector[Nthreads]                  // vetor de threads
2  Inicializar solVector[Nthreads]                 // vetor de soluções
3  Z* = Z_0                                        // melhor solução
4  for k = 0 to Nthreads do
5      solVector[k] = Z_0
6  NotImproved = 0
7  i = 0
8  while (i < Iter) do
9      for k = 0 to Nthreads do
10         THREADCREATE(ILS-CHILD, thVector[k], solVector[k])
11     for k = 0 to Nthreads do                    // sincronização
12         solVector[k] = THREADJOIN(thVector[k])
13     Z = min_k{solVector[k]}                     // melhor mínimo local
14     if f(Z) < f(Z*) then
15         NotImproved = 0
16     else
17         NotImproved = NotImproved + 1
18     if f(Z) ≤ f(Z*) then                        // critério de aceitação
19         Z* = Z
20     if NotImproved ≥ 3 then                     // se não houve melhoras após três iterações
21         for k = 0 to Nthreads do
22             solVector[k] = Z*                   // retornar para Z*
23         NotImproved = 0
24     i = i + 1
25 return Z*
```

Algoritmo 3 Procedimento de busca local – *thread* filha

ILS-CHILD(Z_0)

```
1  Tabu = ∅
2  Z = PERTURBATION(Z_0, 1, Tabu)
3  Z' = LOCALSEARCH(Z, Tabu)
4  return Z'
```

Nossos métodos são baseados em duas fases: a primeira consiste em encontrar uma solução factível e a segunda consiste em melhorar a solução factível.

3.2. Soluções iniciais

Soluções factíveis iniciais são geradas pelo modelo linear inteiro (2)-(6) a seguir:

Definições:

- $A = \{1, ..., na\}$: conjunto de aulas requeridas em R (Definição 1), onde $na = \sum_{r \in R} \theta_r$.
- A_t: conjunto de aulas da turma $t \in T$, $A_t \subset A$;
- A_p: conjunto de aulas do professor $p \in P$, $A_p \subset A$;
- $\widetilde{U}_{pdh} = \begin{cases} 1, & \text{se o professor } p \in P \text{ está disponível no período } h \in H \text{ do dia } d \in D; \\ 0, & \text{caso contrário.} \end{cases}$
- $x_{adh} = \begin{cases} 1, & \text{se a aula } a \in A \text{ é alocada no período } h \in H \text{ do dia } d \in D; \\ 0, & \text{caso contrário.} \end{cases}$

Encontrar x_{adh} (2)

Sujeito a:

$$\sum_{d\in D}\sum_{h\in H} x_{adh} = 1, \qquad \forall\, a \in A \tag{3}$$

$$\sum_{a\in A_t} x_{adh} \leq 1, \qquad \forall\, t \in T,\ d \in D,\ h \in H \tag{4}$$

$$\sum_{a\in A_p} x_{adh} \leq \widetilde{U}_{pdh}, \qquad \forall\, p \in P,\ d \in D,\ h \in H \tag{5}$$

$$x_{adh} \in \{0,1\}, \qquad \forall\, a \in A,\ d \in D,\ h \in H \tag{6}$$

O conjunto de restrições (3) garante que cada aula seja alocada exatamente uma vez. O conjunto (4) garante que cada turma assista uma única aula por período e o conjunto (5) garante que cada professor lecione no máximo uma aula por período e não seja alocado em períodos que está indisponível.

4. Experimentos computacionais

Os métodos foram implementados em C++ e compilados com o g++ 4.8.2. Os experimentos foram realizados no *Linux/Ubuntu 14.04*, rodando em uma máquina virtual *KVM* sobre a distribuição *Linux/CentOS 6*. O sistema hospedeiro roda sobre um servidor com 4 processadores *Intel Xeon* E7-4860 (24MB de *Cache* - 2.26 GHz) e a máquina virtual foi configurada para ocupar 30GB de RAM e 25 núcleos deste hardware. Os experimentos foram realizados com 34 instâncias reais de treze escolas públicas de ensino médio brasileiras. Cada instância foi processada 40 vezes com soluções iniciais geradas pelo *solver* CPLEX 12.6. As violações nas restrições são penalizadas pelos seguintes pesos: $\alpha_3 = \alpha_4 = 100.000$, $\alpha_5 = 100$, $\alpha_6 = 25$, $\alpha_7 = \alpha_8 = \alpha_9 = \alpha_{10} = 10$.

Quatro métodos baseados em ILS foram testados e comparados:

- ILS-TQ: versão sequencial do método proposto em (SAVINIEC et al., 2013), Algoritmo 1;
- ILS-TABU-CI: versão sequencial do Algoritmo 1 empregando lista tabu e movimentos do tipo $\langle slot_i, slot_j, f, index\rangle$;
- ILS-TABU-SI: método ILS-TABU-CI com remoção da componente $index$ dos movimentos;
- ILS-TABU-CI-P: versão *multithreading* do método ILS-TABU-CI, Algoritmos 2 e 3. Definimos um número de *threads* compatível com um PC convencional, $Nthreads = 4$.

As implementações da lista tabu consideram que um movimento $\langle slot_i, slot_j, f, index\rangle = \langle slot_j, slot_i, f, index\rangle$.

Nossa análise compara os métodos em qualidade de soluções e tempo computacional em vários pontos do horizonte de iterações: 500, 750, 1000, 1500 e 2000 iterações. A comparação é baseada em testes estatísticos não-paramétricos utilizando o método de Dunn. Este método realiza comparações entre todos os grupos usando o teste de hipótese de Kruskal-Wallis (KRUSKAL; WALLIS, 1952). O teste de Kruskal-Wallis permite verificar se duas ou mais amostras independentes são originadas de uma mesma população. Isto é, testa-se a hipótese nula "H_0 : *As medianas populacionais são todas iguais*" contra a hipótese alternativa "H_1 : *As medianas não são todas iguais*". Os testes são computados com uma macro *Minitab* implementada por (ORLICH, 2000). Utilizamos um nível de significância de 5%.

Para efetuar as comparações entre os métodos, realizamos uma normalização dos valores de função objetivo e tempo computacional. A normalização é dada por:

$$dr_{ijk} = \frac{\lambda(Z_{ijk})}{\lambda^i_{best}} \tag{7}$$

Onde, λ representa a variável de interesse *função objetivo* ou *tempo computacional* e dr_{ijk} é a distância entre o valor $\lambda(Z_{ijk})$ da j-ésima solução do algoritmo k, na instância i, em relação a $\lambda^i_{best} = \min_{jk}\{\lambda(Z_{ijk})\}$.

4.1. Resultados

Dada a limitação dos métodos exatos na resolução do PPT, há o interesse por heurísticas de baixo custo computacional e com capacidade de gerar soluções primais de boa qualidade. Nossos métodos apresentam bons resultados neste sentido.

O *box-plot* da Figura 2 plota os valores de dr de função objetivo dos métodos em cada ponto analisado no horizonte de iterações. O *box-plot* mostra que o uso de lista tabu com movimentos do tipo $\langle slot_i, slot_j, f, index\rangle$ (método ILS-TABU-CI) melhora a qualidade das soluções, durante as primeiras 1000 iterações, em relação ao método ILS-TQ de (SAVINIEC et al., 2013). Os testes de hipótese apresentados na Tabela 1 dão suporte para garantir esta afirmação. Dado este resultado, analisamos a versão *multithreading* do método ILS-TABU-CI e o ganho em qualidade de soluções é ainda mais expressivo. Notamos que mesmo em 2000 iterações a diferença ainda é significativa.

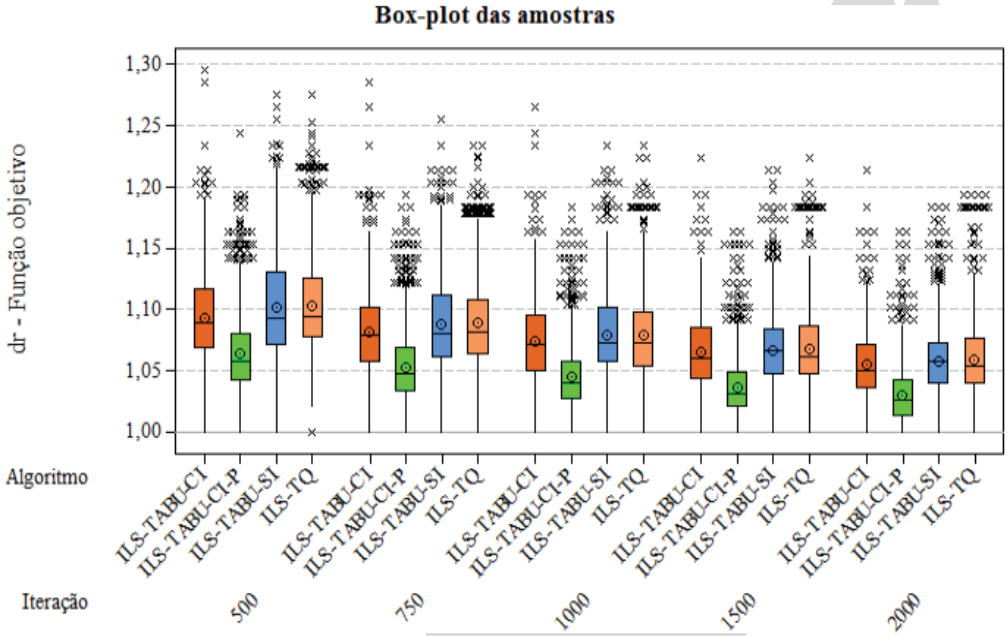


Figura 2: Qualidade das soluções nas iterações 500, 750, 1000, 1500 e 2000.

Ao comparar o tempo de execução em 2000 iterações, Figura 3, observamos que o método ILS-TABU-CI-P é em torno de 50% computacionalmente mais caro que as versões sequenciais. Isto acontece porque o tempo para encontrar um mínimo local não é igual em todas as *threads* filhas e a *thread* mãe precisa esperar a *thread* filha de maior tempo encerrar para efetuar a sincronização (linha 11 do Algoritmo 2). Por outro lado, em 750 iterações, ILS-TABU-CI-P obtêm soluções de qualidade melhor que as soluções encontradas pelas versões sequenciais em 2000 iterações. Esta afirmação é suportada pelos testes de hipótese da Tabela 2, onde os valores de *P-value* nos levam a rejeitar a hipótese nula. Tendo em vista este resultado, o box-plot da Figura 4 plota os intervalos de confiança para as medianas da variável tempo. O box-plot compara o tempo de execução do método

Tabela 1: Pares significativamente diferentes em qualidade de soluções. O melhor método se encontra no lado esquerda da segunda coluna. Em todas as iterações analisadas, a versão *multithreading* é significativamente melhor em qualidade de soluções que as demais versões. Além disso, a versão sequencial ILS-TABU-CI mostrou que o uso de lista tabu combinado com movimentos do tipo $\langle slot_i, slot_j, f, index\rangle$ resultam em melhoras na qualidade das soluções durante as primeiras 1000 iterações.

Iteração	Comparação	P-value
500	ILS-TABU-CI-P vs. ILS-TQ	0,0000
	ILS-TABU-CI-P vs. ILS-TABU-SI	0,0000
	ILS-TABU-CI-P vs. ILS-TABU-CI	0,0000
	ILS-TABU-CI vs. ILS-TQ	0,0000
	ILS-TABU-CI vs. ILS-TABU-SI	0,0026
750	ILS-TABU-CI-P vs. ILS-TQ	0,0000
	ILS-TABU-CI-P vs. ILS-TABU-SI	0,0000
	ILS-TABU-CI-P vs. ILS-TABU-CI	0,0000
	ILS-TABU-CI vs. ILS-TQ	0,0002
	ILS-TABU-CI vs. ILS-TABU-SI	0,0011
1000	ILS-TABU-CI-P vs. ILS-TABU-SI	0,0000
	ILS-TABU-CI-P vs. ILS-TQ	0,0000
	ILS-TABU-CI-P vs. ILS-TABU-CI	0,0000
	ILS-TABU-CI vs. ILS-TABU-SI	0,0001
1500	ILS-TABU-CI-P vs. ILS-TABU-SI	0,0000
	ILS-TABU-CI-P vs. ILS-TQ	0,0000
	ILS-TABU-CI-P vs. ILS-TABU-CI	0,0000
2000	ILS-TABU-CI-P vs. ILS-TQ	0,0000
	ILS-TABU-CI-P vs. ILS-TABU-SI	0,0000
	ILS-TABU-CI-P vs. ILS-TABU-CI	0,0000

Tabela 2: Soluções da versão *multithreading* em 750 iterações versus soluções das versões sequenciais em 2000 iterações. A tabela mostra que a versão *multithreading* é significativamente melhor em qualidade de soluções que as versões sequenciais.

Comparação	P-value
ILS-TABU-CI-P (750) vs. ILS-TABU-CI (2000)	0,0001
ILS-TABU-CI-P (750) vs. ILS-TABU-SI (2000)	0,0000
ILS-TABU-CI-P (750) vs. ILS-TQ (2000)	0,0000

ILS-TABU-CI-P em 750 iterações contra os tempos das versões sequenciais em 2000 iterações e mostra que o tempo de ILS-TABU-CI-P é em torno de 40% menor que o tempo das versões sequenciais.

Para efeito de comparação com trabalhos futuros, a Tabela 3 apresenta os melhores valores de função objetivo (coluna F.O.) e os tempos médios (em segundos) do método ILS-TABU-CI-P nas iterações 750 e 2000, colunas T(750) e T(2000) respectivamente.

5. Conclusões

Em resumo, usando lista tabu combinada com buscas locais *multithreading*, melhoramos significativamente o desempenho do método ILS, tanto em qualidade de soluções como em tempo computacional. Os resultados mostram que esta estratégia gera soluções de melhor qualidade ou equivalente as soluções dos métodos sequenciais em tempo computacional relativamente menor. Isto se deve ao fato do método *multithreading* introduzir maior intensificação na busca, em torno da melhor solução.

Nossos resultados mostram fortes indícios de que estratégias *multithreading* são potenciais ferramentas para melhorar o desempenho de metaheurísticas. Especificamente, métodos sequenciais baseados em *ILS* tem sido aplicado com bastante sucesso na literatura. Sendo assim, trabalhos adicionais podem ser feitos no sentido de testar versões *multithreading* do método em outros problemas, principalmente problemas de *timetabling*.

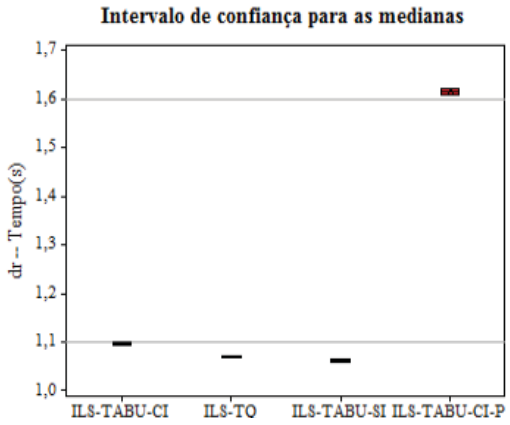


Figura 3: Tempo de execução em 2000 iterações.

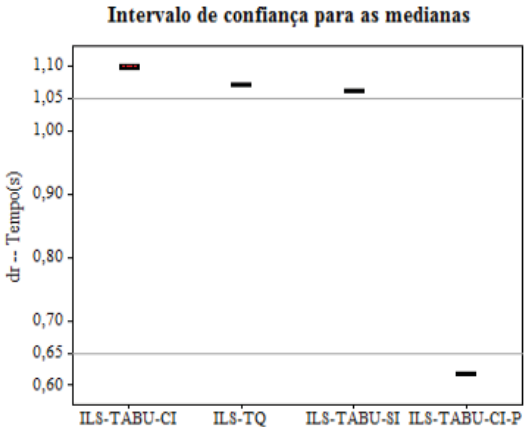


Figura 4: Tempo de execução da versão *multith-reading* em 750 iterações contra os tempos das versões sequenciais em 2000 iterações. Neste gráfico, dr é calculada em relação ao melhor tempo dentre as versões sequenciais.

Tabela 3: Características das instâncias, melhores soluções e tempos médios do método ILS-TABU-CI-P.

ID	$\|T\|$	$\|P\|$	$\sum_{r\in R}\theta_r$	F.O.	T(750)	T(2000)	ID	$\|T\|$	$\|P\|$	$\sum_{r\in R}\theta_r$	F.O.	T(750)	T(2000)
1	12	27	300	980	86,68	226,62	18	5	18	125	460	29,53	77,86
2	12	27	300	980	88,51	230,29	19	4	15	100	360	27,9	73,85
3	13	31	325	1030	91	236,94	20	5	18	125	470	29,98	79,56
4	13	31	325	990	89,11	233,03	21	19	37	475	1550	141,39	365,86
5	9	28	225	870	77,09	203,78	22	12	31	300	1110	82,8	214,59
6	14	29	350	1120	93,46	247,76	23	31	62	775	2760	276,11	707,29
7	20	51	500	1840	170,61	438,56	24	32	75	800	2960	299,2	764,84
8	8	30	200	980	77,26	203,54	25	16	35	400	1400	121,73	315,69
9	13	34	325	1160	88,09	230,13	26	10	21	250	780	68,21	178,69
10	5	17	125	420	29,84	78,37	27	9	20	225	740	59,83	155,9
11	16	35	400	1530	119,92	310,35	28	18	45	450	1630	145,59	381,89
12	16	38	400	1620	106,33	274,69	29	18	44	450	1620	147,44	383,96
13	3	15	75	420	18,15	48,52	30	18	45	450	1650	148,47	388,72
14	16	34	400	1340	113,49	294,16	31	18	45	450	1640	150,3	394,54
15	4	12	100	420	27,7	73,66	32	16	44	400	1450	123,43	322,47
16	8	19	200	640	53,38	140,56	33	16	43	400	1470	127,72	335,75
17	7	21	175	590	45,61	119,94	34	16	43	400	1450	129,07	334,69

6. Agradecimentos

Este trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) – Processo nº 2013/13563-3. Todo o conteúdo apresentado neste material é de inteira responsabilidade dos autores.

Referências

ALVAREZ-VALDÉS, R.; PARREÑO, F.; TAMARIT, J. M. A tabu search algorithm for assigning teachers to courses. *Top*, Springer, v. 10, n. 2, p. 239–259, 2002.

BELLO, G.; RANGEL, M.; BOERES, M. An approach for the class/teacher timetabling problem using graph coloring. *The Practice and Theory of Automated Timetabling VII*, 2008.

BRITO, S. et al. A sa-vns approach for the high school timetabling problem. *Electronic Notes in Discrete Mathematics*, Elsevier, v. 39, p. 169–176, 2012.

COELHO, A.; SOUZA, S. Um algoritmo híbrido baseado em algoritmos meméticos e reconexão por caminhos para resolução do problema de horário escolar. *Anais do SPOLM, Rio de Janeiro*, 2006.

FONSECA, G. H.; SANTOS, H. G. Variable neighborhood search based algorithms for high school timetabling. *Computers & Operations Research*, p. 1–6, 2014. ISSN 0305-0548.

FONSECA, G. H. et al. Goal solver: a hybrid local search based solver for high school timetabling. *Annals of Operations Research*, Springer US, p. 1–21, 2014. ISSN 0254-5330.

GASPERO, L. D.; SCHAERF, A.; MCCOLLUM, B. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Citeseer, 2007.

JACOBSEN, F.; BORTFELDT, A.; GEHRING, H. Timetabling at german secondary schools: tabu search versus constraint programming. In: CITESEER. *Proceedings 6th International Conference on the Practice and Theory of Automated Timetabling (PATAT2006), Brno, Czech Republic*. [S.l.], 2006. p. 439–442.

KOCHETOV, Y.; KONONOVA, P.; PASCHENKO, M. Formulation space search approach for the teacher/class timetabling problem. *The Yugoslav Journal of Operations Research ISSN: 0354-0243 EISSN: 2334-6043*, v. 18, n. 1, p. 1–11, 2008.

KRUSKAL, W. H.; WALLIS, W. A. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, Taylor & Francis, v. 47, n. 260, p. 583–621, 1952.

LEWIS, R.; PAECHTER, B.; MCCOLLUM, B. Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Citeseer, 2007.

LOURENÇO, H.; MARTIN, O.; STÜTZLE, T. Iterated local search. *Handbook of metaheuristics*, Springer, p. 320–353, 2003.

MCCOLLUM, B. et al. The second international timetabling competition: Examination timetabling track. *University of Nottingham, Queen's University, Notingham, Belfast, Technical Report: QUB/IEEE/Tech/ITC2007/Exam/v4.0/17*, 2007.

MOURA, A. et al. Técnicas metaheurísticas aplicadas à construção de grades horárias escolares. *Proceedings of the Brazilian Symposium on Operations Research, Sobrapo, Rio de Janeiro, Brazil*, p. 1319–1330, 2004.

ORLICH, S. *Kruskal-Wallis Multiple Comparisons with a MINITAB Macro Dunn's Test*. [S.l.]: (PDF Document) Minitab, Inc. Available online: http://www.minitab.com/support/macros/default.aspx, 2000.

PILLAY, N. A survey of school timetabling research. *Annals of Operations Research*, Springer, p. 1–33, 2013.

POST, G. et al. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, Springer, v. 194, n. 1, p. 385–397, 2012.

POST, G.; AHMADI, S.; GEERTSEMA, F. Cyclic transfers in school timetabling. *OR Spectrum*, Springer, p. 1–22, 2010.

QU, R. et al. A survey of search methodologies and automated system development for examination timetabling. *Journal of scheduling*, Springer, v. 12, n. 1, p. 55–89, 2009.

SANTOS, H.; OCHI, L.; SOUZA, M. A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. *Journal of Experimental Algorithmics (JEA)*, ACM, v. 10, p. 2–9, 2005.

SANTOS, H. et al. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, Springer, v. 194, n. 1, p. 399, 2012.

SAVINIEC, L. et al. Solving the high school timetabling problem to optimality by using ils algorithms. *Proceedings of the Brazilian Symposium on Operations Research, Sobrapo, Rio de Janeiro, Brazil*, p. 3330–3341, 2013.

SCHAERF, A. Local search techniques for large high school timetabling problems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, IEEE, v. 29, n. 4, p. 368–377, 1999.

SCHAERF, A. A survey of automated timetabling. *Artificial Intelligence Review*, Springer, v. 13, n. 2, p. 87–127, 1999.

SOUSA, V.; MORETTI, A.; PODESTÁ, V. Programação da grade de horário em escolas de ensino fundamental e médio. *Pesquisa Operacional*, SciELO Brasil, v. 28, n. 3, p. 399–421, 2008.

SOUZA, M.; OCHI, L.; MACULAN, N. A grasp-tabu search algorithm for solving school timetabling problems. *Metaheuristics: Computer Decision-Making. Kluwer Academic Publishers, Boston*, p. 659–672, 2003.

SOUZA, M. J. F.; COSTA, F. P.; GUIMARÃES, I. F. G. Um algoritmo evolutivo híbrido para o problema de programação de horários em escolas. *XXII Encontro Nacional de Engenharia de Produção, Curitiba*, 2002.

WILKE, P.; OSTLER, J. Solving the school time tabling problem using tabu search, simulated annealing, genetic and branch & bound algorithms. In: *7th international conference on the practice and theory of automated timetabling, PATAT2008*. [S.l.: s.n.], 2008.

ZHANG, D. et al. A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, Elsevier, v. 203, n. 3, p. 550–558, 2010.