ILS com reconexão de caminhos entre ótimos locais para um problema clássico de escalonamento com antecipação e atraso

Rainer Xavier de Amorim[1], Bruno Raphael Cardoso Dias[1], Rosiane de Freitas Rodrigues[2]

[1]Programa de Pós-Graduação em Informática (PPGI)
[2]Instituto de Computação - Universidade Federal do Amazonas (IComp - UFAM)
Av. Rodrigo Otávio - Campus Universitário, nº 6.200, CEP 69.077-000, Manaus, AM
{rainer,bruno.dias,rosiane}@icomp.ufam.edu.br

RESUMO

Problemas de escalonamento que envolvem critérios de otimização baseados em prazos representam condições importantes do mundo real. Entre esses critérios, os problemas que consideram a combinação das penalidades de antecipação e atraso são o foco de interesse neste trabalho, explorando o conceito *just-in-time* (JIT). Associado a este critério, este trabalho apresenta uma estratégia algorítmica, baseada em buscas locais, que obtém soluções ótimas utilizando a representação de sequência única para o escalonamento e vizinhança baseada em movimentos generalizados de troca de pares, para um problema clássico de escalonamento em ambiente monoprocessado, envolvendo a minimização das penalidades de antecipação e atraso das tarefas. A técnica de reconexão de caminhos é utilizada a fim de se encontrar melhores soluções entre dois ótimos locais. Os experimentos computacionais mostram que a estratégia proposta é competitiva em relação aos resultados existentes na literatura, envolvendo instâncias baseadas no *benchmark* da *OR-library* para casos com 40, 50 e 100 tarefas, propostas por Tanaka (2012).

PALAVRAS CHAVE: algoritmos, penalidades de antecipação e atraso, programação inteira, teoria de escalonamento.
Área principal: Otimização Combinatória.

ABSTRACT

Scheduling problems which involve optimization criteria based on meeting due dates represent important real world conditions. Among these criteria, the problems which consider the combination of earliness and tardiness penalties are the focus of interest in this paper, exploring the just-in-time (JIT) concept. Associated to this criterion, this paper presents an algorithmic strategy, based on two local searches, which obtain optimal solutions using the single sequence representation for scheduling and neighborhood based on Generalized Pairwise Interchange moves, for a classical scheduling problem on a single machine, involving the minimization of the earliness and tardiness penalties of the jobs. The path-relinking technique is used in order to find better solutions between two local optima. The computational experiments shows that our strategy is competitive in relation to the existing results from the literature, involving instances from the OR-library benchmark for cases with 40, 50 and 100 jobs, proposed by Tanaka (2012).

KEYWORDS: algorithms, earliness and tardiness penalties, integer programming, scheduling theory.
Main area: Combinatorial Optimization.

1. Introdução

Problemas de escalonamento que envolvem restrições de antecipação (*earliness*) e atraso (*tardiness*) são uma das principais vertentes de pesquisa sobre a classe geral de escalonamento. O estudo destes problemas foi motivado pela adoção do conceito *Just-in-Time* (JIT) na indústria, que consiste em completar uma tarefa no prazo exato de entrega. No entanto, existem muitas outras aplicações que usam antecipação e minimização de atraso [Pinedo (2012)]. Problemas de JIT ocorrem comumente na produção de itens perecíveis, tais como: a colheita de itens de origem vegetal, que deve ser feita quando o item estiver maduro, e itens como leite e carne fresca, para evitar que haja uma possível deterioração e também para que o produtor não perca o prazo de entrega, dentre outros. Este tipo de conceito pode ser representado como uma linha de produção em uma fábrica onde todos os componentes de algum item devem estar prontos ao mesmo tempo, ou seja, pode-se representar qualquer sistema no qual várias atividades devem ser completadas ao mesmo tempo [Gordon (2002)].

Neste contexto, esses problemas são formalizados como segue: seja $j = \{1,...,n\}$ um conjunto de tarefas a serem processadas em um ambiente monoprocessado, sem preempção. A máquina pode processar pelo menos uma tarefa, sendo uma por vez, e todas as tarefas devem ser processadas. Cada tarefa j possui um tempo de processamento positivo (*processing time*), denotado por p_j, um prazo de termino sugerido (*due date*) d_j e um peso positivo (*weight*) w_j. A antecipação de uma tarefa j, de acordo com o seu prazo de término, é definida como $E_j = max\{0, d_j - C_j\}$, por outro lado, o atraso de uma tarefa j, de acordo com o seu prazo de término, é definido como $T_j = max\{0, C_j - d_j\}$, onde C_j é o tempo de completude de uma tarefa [Brucker (2006), Rodrigues (2009)].

O problema de escalonamento considerado neste trabalho consiste em escalonar as tarefas em uma máquina com o objetivo de minimizar o $\sum_{j=1}^{n} \alpha_j E_j + \sum_{j=1}^{n} \beta_j T_j$, sendo α_j e β_j, pesos positivos e diferenciados para antecipação e atraso, respectivamente. Este problema é referenciado como $1||\sum \alpha_j E_j + \sum \beta_j T_j$, na notação de três campos proposta por Graham et al. (1979), e trata-se de um problema NP-Difícil, pois faz parte de uma generalização do problema $1|d_j|\sum T_j$, provado ser NP-Difícil por Du e Leung (1990).

Devido a essa dificuldade inerente em resolver o problema, existem muitas pesquisas envolvendo métodos heurísticos para a obtenção de boas soluções (ótimas em alguns casos) para as instâncias do problema. Uma das técnicas envolve busca local iterada (*Iterated Local Search* - ILS), conforme visto em Rodrigues et al.(2008) que consiste em, a partir de uma solução inicial, encontrar o ótimo local de uma vizinhança previamente definida e, posteriormente, efetuar perturbações (alterações aleatórias) na solução de forma a visitar outras partes do espaço de busca. No entanto, como o espaço de soluções tem tamanho exponencial, é interessante que seja possível aproveitar a vizinhança da melhor forma possível. Neste trabalho, o algoritmo ILS original foi modificado de forma a adotar um procedimento de reconexão de caminhos (*Path-Relinking* - PR), técnica originalmente proposta para a diversificação do algoritmo *scatter search* [Glover et al. (2000)]. Assim, é possível convergir mais rapidamente às melhores soluções e também obter soluções melhores em instâncias de tamanho grande.

Este trabalho é organizado da seguinte forma: na seção 2 serão apresentados os trabalhos relacionados da literatura que tratam de antecipação e atraso, na seção 3 é apresentado o algoritmo de busca local considerado na implementação da técnica de reconexão

de caminhos para o problema de escalonamento *JIT*, com penalidades diferenciadas para tarefas antecipadas e atrasadas, apresentado na seção 4. Na seção 5 são apresentados os experimentos computacionais e a última seção é a conclusão.

2. Trabalhos relacionados

Nesta seção serão detalhados os trabalhos relacionados da literatura que abordam problemas de escalonamento com penalidades de antecipação, atraso e a forma combinada de ambos (JIT), e também serão apresentadas as técnicas e métodos heurísticos e exatos propostos na literatura para solucionar os problemas em questão em ambiente monoprocessado.

2.1. Escalonamento com Atraso de Tarefas

Minimizar o atraso total de uma tarefa é uma das considerações mais importantes na produção industrial, especialmente quando se tem prazos a serem cumpridos. Na literatura, abordagens exatas para este problema são, em sua maioria, baseadas em algoritmos de programação dinâmica e *Branch-and-Bound*. Abordagens aproximadas são baseadas em regras de despacho de tarefas e metaheurísticas.

A seguir serão apresentados alguns problemas de escalonamento com penalidades de atraso em ambiente monoprocessado, assim como as principais abordagens de resolução presentes na literatura. A Figura 1 mostra onde é possível obter o valor de atraso de uma tarefa na linha do tempo, pode-se observar também que antes do prazo d_j, o valor de atraso é nulo e, após esse prazo, o valor de atraso tende a crescer.

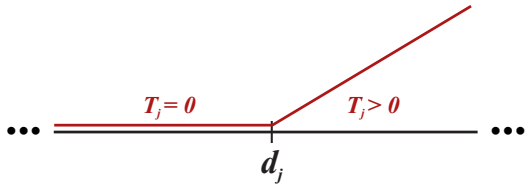

Figura 1. Definição do valor de atraso na linha do tempo.

Tan and Narasimhan (1997) propuseram um algoritmo de recozimento simulado (*simulated annealing*) e demonstrou por análise computacional que este algoritmo é melhor que os já presentes na literatura. Trabalhos relacionados com algoritmos genéticos e tempos de preparação (*setups*) podem ser encontrados com abordagens diferentes em Tan et al. (2000) e França et al. (2001). Mendes et al. (2002) implementaram um procedimento conhecido como *Multi Start*, que trata-se de um algoritmo que cria uma solução inicial aleatória e depois aplica uma busca local a fim de encontrar uma solução ótima na vizinhança do espaço de soluções.

Tian et al. (2006) forneceram um algoritmo de tempo $O(n^2)$ para solucionar o problema $1|r_j, p_j = p, pmtn|\sum T_j$ usando o conceito de blocos (técnica de decomposição), conceito onde o problema de escalonamento é decomposto em dois subproblemas, onde cada subproblema envolve um escalonamento ótimo, que é então distribuído para cada bloco. Finalmente, é investigado o escalonamento de um bloco com tarefas de tempos iguais e algumas propriedades são derivadas para a montagem de um algoritmo para solucionar o problema.

Pinedo (2012) propôs um algoritmo aproximado que fornece, em tempo polinomial, uma solução quase ótima para o problema $1||\sum T_j$, mas tal problema é NP-difícil, portanto, nem um algoritmo *Branch-and-Bound* e nem um algoritmo de programação dinâmica podem fornecer soluções ótimas em tempo polinomial.

Existem na literatura abordagens para o ambiente de máquinas paralelas idênticas, mas que também consideram o ambiente monoprocessado, como pode ser observado em Rodrigues et al. (2008), onde foi proposto um algoritmo heurístico para o problema $P||\sum w_j T_j$ em que o escalonamento é definido como uma sequência única de tarefas em uma máquina (Figura 2 (a)) e, para a versão em máquinas paralelas idênticas, cada tarefa aparece apenas em uma sequência de máquinas. Se todas as tarefas nesta determinada sequência forem processadas, a máquina é considerada ociosa.

A Figura 2 (b) mostra representação de um escalonamento em máquinas paralelas idênticas através do **gráfico de Gantt**. Para esse mesmo problema de escalonamento (considerando somente máquinas paralelas idênticas), Groce et al. (2012) apresentaram uma heurística melhorada cujos resultados computacionais apresentados são melhores que os resultados existentes na literatura, incluindo os apresentados por Rodrigues et al. (2008). Um algoritmo exato pode ser observado em Pessoa et al. (2010), onde foi proposto um método baseado em *Branch-and-Cut-and-Price* para o problema $P||\sum w_j T_j$, também considerando o ambiente monoprocessado e máquinas paralelas idênticas.

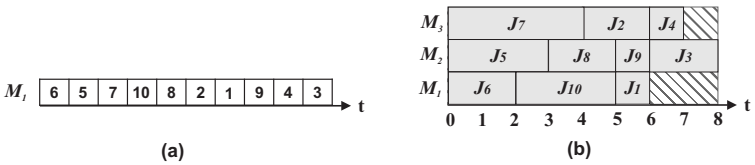

Figura 2. (a) Representação em sequência única. (b) Representação em máquinas paralelas.

2.2. Escalonamento com Antecipação de Tarefas

Nesta seção serão apresentados alguns problemas de escalonamento com penalidades de antecipação em ambiente monoprocessado. Em ambientes reais, os fabricantes não costumam estipular um prazo para uma tarefa específica; ao invés disso, eles possuem datas pré-definidas ou previstas em que suas tarefas devem ser finalizadas, um exemplo seria as tarefas que devem ser entregues no final de cada dia de trabalho cuja antecipação é definida como a diferença entre o tempo de entrega e o tempo de completude (C_j) dessa tarefa [Yang (2000)]. A Figura 3 mostra onde é possível obter o valor de antecipação de uma tarefa na linha do tempo, pode-se observar também que após o prazo d_j, o valor de antecipação é nulo e, antes desse prazo, o valor de antecipação tende a crescer.

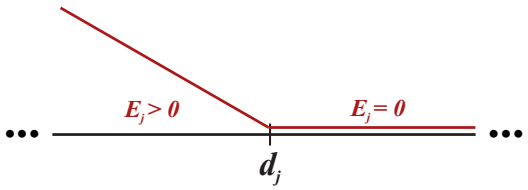

Figura 3. Definição do valor de antecipação na linha do tempo.

Cheng et al. (1996) estudaram o problema de escalonamento em lotes (*batches*), onde as tarefas em um lote são entregues ao cliente juntamente com o tempo de completude

da última tarefa no lote, a antecipação dessa tarefa é definida como a diferença entre o tempo de entrega do lote, em que essa tarefa está inserida, e o seu tempo de completude. A abordagem utilizada foi a de programação dinâmica para o escalonamento de máquinas paralelas e uma abordagem semelhante foi considerada para o problema de escalonamento em lotes para o ambiente monoprocessado.

Gordon e Strusevich (1999) consideraram o ambiente monoprocessado com prazos e problemas de escalonamento com n tarefas em que os prazos são obtidos de tempos de processamento com folgas q. Eles fazem uma consideração para as seguintes funções objetivo (importantes para o escalonamento e controle de aplicações): antecipação ponderada $\sum_{j=1}^{n} w_j E_j$, onde w_j é um dado peso da tarefa j que indica sua relativa importância e; antecipação com ponderações exponenciais $\sum_{j=1}^{n} w_j exp(\gamma E_j)$, onde $\gamma \neq 0$.

Koksalan et al. (1998) consideraram o problema de escalonamento bicritério e máxima antecipação de tarefas em ambiente monoprocessado, eles desenvolveram uma heurística para gerar sequências eficientes para o caso em que as máquinas possuem tempo ocioso e também fizeram considerações para casos sem tempo ocioso. Um caso de problemas de escalonamento com entrega em lotes e penalidades de antecipação foi estudado por Yang (2000), onde foram considerados os problemas de escalonamento que minimizam dois tipos de penalidades de antecipação: um deles é achar a sequência ótima tal que o tempo obtido no somatório das antecipações ponderadas seja mínimo; e o outro problema é achar a sequência ótima tal que o valor obtido no somatório das antecipações ponderadas seja mínimo.

Zhao e Tang (2007) consideraram o problema de escalonamento em ambiente monoprocessado com efeitos tardios, eles propuseram soluções polinomiais para a minimização do problema de *makespan* (momento em que termina a execução da última tarefa), o problema de minimização da soma dos tempos de completude e a soma de tarefas antecipadas ponderadas.

2.3. Escalonamento com Antecipação e Atraso

Nesta seção, serão apresentados problemas de escalonamento com penalidades de antecipação e atraso em ambiente monoprocessado. Baker e Scudder (1990) apresentaram uma revisão bibliográfica com um foco em problemas que envolvem penalidades de antecipação e atraso. Uma classificação e variações para problemas de escalonamento que envolvem prazos em comum também pode ser consultada em Gordon et al. (2002).

A Figura 4 mostra onde é possível obter o valor de JIT de uma tarefa na linha do tempo, o que se quer é exatamente minimizar o valor de antecipação e atraso de modo que se tenha uma tarefa o mais próxima possível do prazo d_j, dessa forma, quanto mais próximo a tarefa estiver do prazo estipulado, menor será o seu valor de JIT.

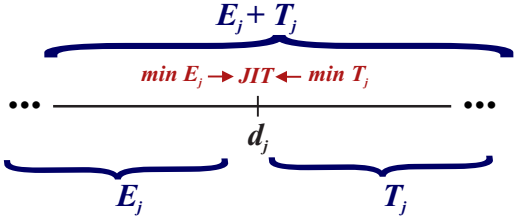

Figura 4. Definição do valor de *just-in-time* (JIT) na linha do tempo.

Segundo Pinedo (2012), a função objetivo que envolve a minimização de atraso pertence as funções objetivo que possuem medidas **regulares** de desempenho. Uma função objetivo possui uma medida de desempenho regular quando é não-decrescente em $C_1, ..., C_n$. Pesquisas recentes consideram o estudo de funções objetivo que são **não-regulares**, são aquelas cuja tarefa j possui um prazo d_j e que pode estar sujeito à penalidades de antecipação.

Dessa forma, a penalidade de antecipação é não-crescente em C_j. Assim como a soma total de antecipação e atraso, ou seja, $\sum_{j=1}^{n} E_j + \sum_{j=1}^{n} T_j$, que é portanto, não-regular. Outra função objetivo geral para esse problema, que também é não-regular, é a função objetivo que envolve a soma total de antecipação ponderada mais o atraso ponderado das tarefas, isto é, $\sum_{j=1}^{n} \alpha_j E_j + \sum_{j=1}^{n} \beta_j T_j$.

Shabtay e Steiner (2012) apresentam uma revisão da literatura sobre problemas de escalonamento JIT, onde é apresentado o problema de maximização do número ponderado de tarefas que são finalizadas exatamente no prazo e também são apresentados vários ambientes de escalonamento com tempos de processamento fixos e variáveis.

Kedad-Sidhoum et al. (2008) propuseram duas novas famílias de limites inferiores para o problema de escalonamento em que as tarefas possuem prazos distintos com custos de antecipação e atraso. Primeiro foi feita uma generalização de duas atribuições de limites inferiores para o problema com máquinas monoprocessadas e máquinas paralelas, e segundo, foi investigado uma formulação indexada por tempo para o problema a fim de se obter eficientes limites inferiores através da geração de colunas e relaxação lagrangeana. Para o limite superior, foi apresentado o algoritmo de busca local.

Um algoritmo *Branch-and-Bound* para o problema de escalonamento com penalidades de antecipação e atraso pode ser visto em Tanaka et al. (2003), onde o procedimento começa com uma solução inicial construída pela heurística de troca de pares adjacentes (*Adjacent Pairwise Interchange* - API), e depois é executado um algoritmo de sequência de tarefas do menor para o maior prazo (*Earliest Due Date* - EDD) para sequenciar e efetuar movimentos API até que a função objetivo não possa mais ser melhorada.

3. Algoritmo heurístico de busca local iterada (ILS)

O algoritmo de busca local iterada utilizado neste trabalho foi uma adaptação do algoritmo proposto por Rodrigues et al. (2008) para o problema de escalonamento $P||\sum w_j T_j$. A ideia principal dessa heurística é representar um escalonamento mínimo para o problema com atraso total ponderado de tarefas como uma sequência única de tarefas. A Figura 2 (a) mostra a representação em sequência única de tarefas utilizada pelo algoritmo, que trata-se somente de uma permutação do conjunto de tarefas para o ambiente monoprocessado. O algoritmo possui bom desempenho, obtendo boas soluções para instâncias com 40 e 50 tarefas. A seguir serão considerados os detalhes da adaptação dessa heurística de busca local para o problema de escalonamento com antecipação e atraso.

Dada uma permutação inicial de π tarefas, $\pi = (\pi_1, \pi_2, ..., \pi_n)$, o algoritmo executa uma busca local na vizinhança da solução atual definida pelos movimentos generalizados de troca de pares (*Generalized Pairwise Interchange* - GPI) que, por sua vez, é definido por dois tipos de movimentos, o primeiro deles é o movimento de troca (*swap*), que trata da troca de posições de duas tarefas (não necessariamente adjacentes) em π, Figura 5 (a); e o segundo deles é o movimento de remoção (*move*), que trata da remoção de uma tarefa

em π seguida pela sua reinserção na sequência em uma nova posição, Figura 5 (b). Cada busca local é realizada até que não seja mais possível melhorar a solução corrente.

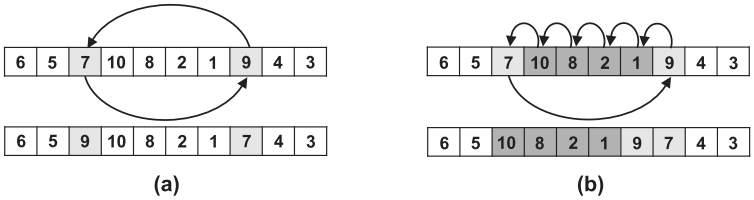

Figura 5. Exemplo de movimentos generalizados de troca de pares: (a) movimento de troca, onde a posição das duas tarefas i e j na sequência é trocada e (b) movimento de remoção, onde a tarefa i é removida de sua posição original e inserida na posição da tarefa j, a subsequência de tarefas entre as tarefas i e j são reposicionadas para dar espaço a tarefa i na sua nova posição.

Durante a busca local, a mudança é aceita somente quando a nova solução é melhor que a solução corrente. Entretanto, soluções ótimas locais geralmente possuem muitas soluções vizinhas com o mesmo custo. Dessa forma, para impedir que o algoritmo continue efetuando a busca por soluções melhores, é necessário que se tenha um critério de desempate de soluções que define, dentre outras soluções de mesmo valor de função objetivo, qual deve ser considerada a melhor solução. De modo formal, dada uma sequência $\pi = (\pi_1, ..., \pi_n)$, define-se $b(\pi)$ como uma pontuação da permutação π, que será dada por: $b(\pi) = \sum_{j=1}^{n} d_{\pi_j} \cdot (n - j + 1)$.

Assim, dada uma sequência de tarefas π e ρ com o mesmo custo, se $b(\rho) < b(\pi)$, então a solução ρ é considerada a melhor solução. O critério de desempate considera que as tarefas que tem o menor prazo sejam escalonadas primeiro, esse critério é baseado no algoritmo de sequência de tarefas do menor para o maior prazo, que gera uma solução ótima para o problema $1||\sum \alpha_j E_j + \sum \beta_j T_j$, ou seja, quando há uma solução sem tarefas antecipadas e com atraso.

4. ILS com reconexão de caminhos

Esta seção apresenta uma estratégia algorítmica baseada em buscas locais, utilizando uma adaptação da busca local iterada proposta por Rodrigues et al. (2008), a fim de pesquisar mais do espaço solução. Em vez de se trabalhar com apenas com uma sequência única (que codifica uma solução), foram utilizadas duas sequências únicas em cada iteração. Antes da primeira iteração, é realizada uma permutação aleatória na sequência única seguindo a regra de sequência de tarefas do menor para o maior prazo, a fim de que se tenha duas sequências iniciais viáveis. Em seguida, o algoritmo realiza uma busca local sobre as duas soluções viáveis. Após a busca local, tem-se duas soluções ótimas locais, mas podem haver soluções melhores no espaço de soluções existente entre essas duas soluções.

A partir daí, a técnica de reconexão de caminhos é então utilizada para explorar o espaço entre as duas soluções iniciais encontradas. Na implementação proposta, optou-se por utilizar a técnica de reconexão de caminhos reversos, que parte da **melhor solução** para a **pior solução** (*Backward Path-Relinking*) onde, a solução de partida é a melhor solução das duas soluções iniciais encontradas e a **solução guiada** ou **solução de destino** é a pior. A melhor solução encontrada no caminho é armazenada e, finalmente, as três soluções (duas soluções obtidas por buscas locais e uma solução obtida pela reconexão de caminhos) são comparadas entre si e entre a melhor solução global encontrada (solução corrente do

algoritmo). A melhor solução entre as quatro soluções candidatas é então considerada a melhor nova solução global, como pode ser observado no esquema da Figura 6.

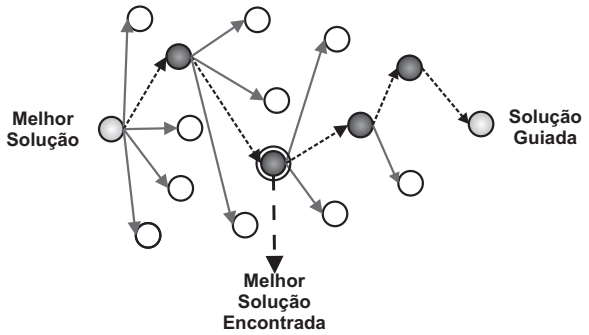

Figura 6. Esquema para o algoritmo de reconexão de caminhos reversos (*Backward Path-Relinking*), onde as soluções mais escuras, e com as setas rachuradas, representam o caminho percorrido de melhores soluções no espaço de soluções (representado pela cor branca) entre as duas soluções iniciais, que por sua vez são representadas pelas soluções na cor cinza claro.

Para implementar a técnica de reconexão de caminhos, primeiramente guarda-se a solução inicial, que é a melhor solução das duas soluções iniciais encontradas. Esta solução, que será chamada de **caminho de sequência**, será alterada para se tornar igual à solução guiada. O algoritmo então verifica, para cada posição do caminho de sequência, uma solução que, quando alterada, se torne igual a solução guiada, retornando assim a melhor solução.

Dessa forma, para fazer a posição j no caminho de sequência θ se tornar igual à posição j da solução guiada π, deve-se encontrar a posição k do elemento π_i no caminho de sequência. Encontrada a posição k, os elementos θ_i e θ_j são trocados no caminho de sequência. Assim, quando o melhor movimento é encontrado no caminho de sequência, e quando a posição do caminho de sequência se torna igual a posição da solução guiada, essa posição é marcada como fixa e não será mais modificada. Este procedimento é repetido até que todas as posições se tornem fixas (n iterações).

Como a reconexão de caminhos explora uma boa porção do espaço de busca entre dois ótimos locais, o passo de perturbação da solução é removido de forma a sempre tentar seguir para uma nova e extensa área do espaço de soluções que pode ser explorada por esses procedimentos (duas buscas locais e reconexão de caminhos). O Pseudocódigo para este algoritmo melhorado integrando busca local iterada e reconexão de caminhos é apresentado no Algoritmo 1 e o pseudocódigo que detalha a técnica de reconexão de caminhos é apresentado no Algoritmo 2.

5. Experimentos computacionais

Para testar o desempenho do algoritmo, existem instâncias disponíveis na *OR-Library* para o problema $1||\sum w_j T_j$ e também as disponibilizadas por Tanaka (2012) para os problemas $1||\sum w_j T_j$ e $1||\sum \alpha_j E_j + \sum \beta_j T_j$ com e sem tempo ocioso. As instâncias utilizadas para os experimentos computacionais deste trabalho foram retiradas de Tanaka (2012), que se baseou nas instâncias disponibilizadas na *OR-Library* para o problema de escalonamento com penalidades de atraso somente.

O algoritmo proposto neste trabalho conseguiu alcançar os valores ótimos, disponibilizados por Tanaka (2012), para todos os casos testados. As tabelas 1, 2 e 3 mostram os

Algoritmo 1: ILS para o problema $1||\sum \alpha_j E_j + \sum \beta_j T_j$ com reconexão de caminhos reversos.

```
1  i ← 1;
2  π ← uma permutação seguindo a regra do menor para o maior prazo;
3  θ ← uma permutação randômica de π;
4  π* ← melhor solução corrente;
5  enquanto i < N faça
6      se i > 0 então
7          π e θ ← permutações randômicas de π e θ;
8      fim se
9      Aplica movimentos generalizados de troca de pares em π e em θ, até que nenhuma melhora seja possível;
10     se w(π) < w(θ) então
11         γ ← Reconexão-de-Caminhos(π, θ);
12     fim se
13     senão
14         γ ← Reconexão-de-Caminhos(θ, π);
15     fim se
16     se w(π) < w(θ) e w(π) < w(γ) então
17         se w(π) < w(π*) então
18             π* ← π;
19         fim se
20         senão
21             se w(θ) < w(π) e w(θ) < w(γ) então
22                 se w(θ) < w(π*) então
23                     π* ← θ;
24                 fim se
25                 senão
26                     se w(γ) < w(π*) então
27                         π* ← γ;
28                     fim se
29                 fim se
30             fim se
31         fim se
32     fim se
33     i ← i + 1;
34 fim enquanto
```

experimentos computacionais para as instâncias de 40, 50 e 100 tarefas, respectivamente, em ambiente monoprocessado, onde é possível observar os valores ótimos da literatura e os valores obtidos pelo algoritmo proposto juntamente com os tempos de processamento em segundos que, para algumas instâncias, o algoritmo proposto possui um desempenho melhor que o da literatura. O desafio agora consiste em realizar experimentos envolvendo instâncias com um número maior de tarefas. Os resultados foram obtidos a partir da execução em um computador com um processador Intel Core i5 2.66 GHz, 4GB de RAM e com o sistema operacional Ubuntu.

6. Conclusões

Neste trabalho foi abordado o problema de escalonamento em máquinas monoprocessadas e tarefas independentes, envolvendo a minimização conjunta das penalidades de antecipação e atraso das tarefas a serem escalonadas. Foi apresentada uma estratégia algorítmica baseada em buscas locais, onde foi realizada uma adaptação da heurística de busca local iterada proposta por Rodrigues et al. (2008) para problemas de escalonamento envolvendo penalidade de atraso.

Uma estratégia envolvendo a heurística de busca local adaptada e a técnica de reconexão de caminhos foi desenvolvida. O algoritmo proposto neste trabalho conseguiu atingir os valores ótimos disponibilizados por Tanaka (2012), para todos os casos testados. Os experimentos computacionais mostram que o algoritmo proposto consegue atingir valores ótimos com um desempenho melhor que o da literatura para algumas instâncias.

Algoritmo 2: Reconexão-de-Caminhos(Início,SoluçãoGuiada).

Entrada: Início,SoluçãoGuiada
Saída: MelhorEncontrado

```
1  Caminho ← Início;
2  Contador ← 1;
3  MelhorSoluçãoEncontrada ← ∞;
4  enquanto Contador ≤ n faça
5      MelhorSoluçãoAlterada ← ∞;
6      enquanto j ≤ n faça
7          se j não é fixo então
8              k ← posição da SoluçãoGuiada_j no Caminho;
9              Troca Caminho_j e Caminho_k;
10             se w(caminho) < MelhorSoluçãoAlterada então
11                 MelhorJ ← j;
12                 MelhorK ← k;
13                 MelhorSoluçãoAlterada ← w(caminho);
14             fim se
15             Troca Caminho_k e Caminho_j;
16         fim se
17     fim enquanto
18     Troca Caminho_MelhorJ e Caminho_MelhorK;
19     se w(caminho) < MelhorSoluçãoEncontrada então
20         MelhorEncontrado ← Caminho;
21         MelhorSoluçãoEncontrada ← w(caminho);
22     fim se
23     Marca j como fixo;
24     Contador ← Contador +1;
25 fim enquanto
```

O desafio agora consiste em realizar experimentos envolvendo instâncias com um número maior de tarefas. Ainda como possíveis extensões, deseja-se adaptar a estratégia algorítmica exata proposta por Pessoa et al. (2010) para o problema de escalonamento com atraso ponderado, bem como elaborar um método híbrido envolvendo técnicas de corte.

Agradecimentos

Agradecemos ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro.

Referências

Baker, K. R. e Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: a review. *Oper. Res.*, 38:22–36.

Brucker, P. (2006). Scheduling algorithms. *Springer Publishing Company, Incorporated*, 5a ed.:1–104.

Cheng, T., Gordon, V. S., e Kovalyov, M. Y. (1996). Single machine scheduling with batch deliveries. *European Journal of Operational Research*, 94:277–283.

Croce, F. D., Garaix, T., e Grosso, A. (2012). Iterated local search and very large neighborhoods for the parallel-machines total tardiness problem. *Computers & Operations Research*, 39:1213–1217.

Du, J. e Leung, J. Y. (1990). Minimizing total tardiness on one machine is NP-Hard. *Mathematics of Operations Research*, 15:483–495.

França, P. M., Mendes, A., e Moscato, P. (2001). A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research*, 132:224 – 242.

Glover, F., Laguna, M., e Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684.

Gordon, V., Proth, J.-M., e Chu, C. (2002). A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, 139:1–25.

Gordon, V. S. e Strusevich, V. A. (1999). Earliness penalties on a single machine subject to precedence constraints: SLK due date assignment. *Computers & OR*, 26:157–177.

Graham, R. L., Lawler, E. L., Lenstra, J. K., e Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326.

Kedad-Sidhoum, S., Solis, Y. R., e Sourd, F. (2008). Lower bounds for the earliness-tardiness scheduling problem on parallel machines with distinct due dates. *European Journal of Operational Research*, 189:1305–1316.

Koksalan, M., Azizoglu, M., e Kondakci, S. K. (1998). Minimizing flowtime and maximum earliness on a single machine. *IIE Transactions*, 30:192–200.

Mendes, A., Franca, P., e Moscato, P. (2002). Fitness landscapes for the total tardiness single machine scheduling problem. *Neural Network World*, 2:165–180.

Pessoa, A., Uchoa, E., de Aragão, M. P., e Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, 2:259–290.

Pinedo, M. L. (2012). Scheduling: Theory, algorithms, and systems. *Springer Publishing Company, Incorporated*, 4a ed.:1–104.

Rodrigues, R. (2009). Caracterizações e algoritmos para problemas clássicos de escalonamento. *COPPE – Programa de Engenharia de Sistemas e Computação - Universidade Federal do Rio de Janeiro*, PhD thesis:1–22.

Rodrigues, R., Pessoa, A., Uchoa, E., e de Aragão, M. P. (2008). Heuristic algorithm for the parallel machine total weighted tardiness scheduling problem. *Relatórios de pesquisa em engenharia de produção*, 8:1–12.

Shabtay, D. e Steiner, G. (2012). Scheduling to maximize the number of just-in-time jobs: A survey. *Springer Optimization and Its Applications*, 60:3–20.

Tan, K. e Narasimhan, R. (1997). Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach. *Omega*, 25:619–634.

Tan, K.-C., Narasimhan, R., Rubin, P. A., e Ragatz, G. L. (2000). A comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times. *Omega*, 28:313–326.

Tanaka, S. (2012). An exact algorithm for the single-machine earliness-tardiness scheduling problem. *Springer Optimization and Its Applications*, 60:21–40.

Tanaka, S., Sasaki, T., e Araki, M. (2003). A branch-and-bound algorithm for the single-machine weighted earliness-tardiness scheduling problem with job independent weights. *Systems, Man and Cybernetics*, 2:1571–1577.

Tian, Z., Ng, C. T., e Cheng, T. C. E. (2006). An $O(n^2)$ algorithm for scheduling equal-length preemptive jobs on a single machine to minimize total tardiness. *Journal of Scheduling*, 9:343–364.

Yang, X. (2000). Scheduling with generalized batch delivery dates and earliness penalties. *IIE Transactions*, 32:735–741.

Zhao, C.-L. e Tang, H.-Y. (2007). Single machine scheduling problemas with an aging effect. *J. Appl. Math and Computing*, 25:305–314.

Tabela 1. Resultados para $1||\sum \alpha_j E_j + \sum \beta_j T_j$ com 40 tarefas e 1 máquina usando ILS+PR.

Instância	Ótimo Tanaka (2012)	Obtido ILS+PR	Iterações ILS+PR	Tempo Tanaka (2012)	Tempo ILS+PR
1	54640	54640	2	0.08	0.05
11	29924	29924	0	0.09	0.01
21	77819	77819	1	0.12	0.03
31	23291	23291	0	0.10	0.02
41	59093	59093	0	0.10	0.02
51	47667	47667	0	0.15	0.02
61	21737	21737	0	0.11	0.01
71	90521	90521	0	0.14	0.02
81	12552	12552	3	0.15	0.09
91	48311	48311	0	0.15	0.03
101	85961	85961	15	0.12	0.38
111	32374	32374	3	0.11	0.10
121	122538	122538	3	0.10	0.09

Tabela 2. Resultados para $1||\sum \alpha_j E_j + \sum \beta_j T_j$ com 50 tarefas e 1 máquina usando ILS+PR.

Instância	Ótimo Tanaka (2012)	Obtido ILS+PR	Iterações ILS+PR	Tempo Tanaka (2012)	Tempo ILS+PR
1	130548	130548	19	0.20	0.78
11	54167	54167	6	0.18	0.35
21	214555	214555	2	0.33	0.11
31	37565	37565	16	0.23	0.72
41	71949	71949	0	0.28	0.04
51	56208	56208	0	0.15	0.04
61	34640	34640	119	0.49	5.80
71	150978	150978	1	0.28	0.09
81	17433	17433	0	0.31	0.05
91	89715	89715	38	0.28	1.73
101	90880	90880	3	0.19	0.19
111	30170	30170	2	0.27	0.12
121	79007	79007	0	0.17	0.04

Tabela 3. Resultados para $1||\sum \alpha_j E_j + \sum \beta_j T_j$ com 100 tarefas e 1 máquina usando ILS+PR.

Instância	Ótimo Tanaka (2012)	Obtido ILS+PR	Iterações ILS+PR	Tempo Tanaka (2012)	Tempo ILS+PR
1	405122	405122	98	3.49	36.73
11	252623	252623	377	3.80	139.60
21	898927	898927	0	5.63	0.32
31	193480	193480	277	4.68	113.99
41	463554	463554	23	3.88	9.77
51	444991	444991	51	2.91	20.33
61	102185	102185	65	2.65	28.27
71	640932	640932	9	3.13	3.59
81	47629	47629	18	2.55	8.22
91	249743	249743	779	6.75	294.96
101	356397	356397	34	6.05	15.49
111	161642	161642	116	8.22	56.97
121	471761	471761	113	3.93	49.47