Escalonamento de Tarefas com Níveis Variados de Criticidade

Elisa Dell'Arriva[1], **Flávio K. Miyazawa**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brazil

Abstract. *We address the problem of nonpreemptively scheduling unit length tasks with different criticality levels on multiple machines of a given capacity. If necessary at runtime, a task can run for extra time units, depending on its criticality, while less critical tasks may be cancelled. A task can never cancel more critical tasks. The objective is to minimize the number of machines needed to schedule all tasks. We present an* APTAS *for the case where every task is significantly critical, i.e., there is a minimum threshold on the criticality values.*

Resumo. *Abordamos o problema de escalonamento não preemptivo de tarefas unitárias com diferentes níveis de criticidade em múltiplas máquinas com capacidade definida. Se necessário durante a execução, uma tarefa pode executar por um tempo extra, que depende de sua criticidade, e tarefas menos críticas podem ser canceladas. O objetivo é minimizar o número de máquinas necessárias para escalonar todas as tarefas. Apresentamos um* APTAS *para o caso em que todas as tarefas são significativamente críticas, ou seja, há um limite mínimo para os valores de criticidade.*

1. Introdução

Em problemas de escalonamento, de forma geral, são dados um conjunto de tarefas, seus tempos de execução e um conjunto de máquinas; queremos alocar as tarefas nas máquinas otimizando algum recurso. Uma solução é uma atribuição de instantes de início para cada tarefa, que chamamos de *escalonamento*. As variantes são muitas. Temos, por exemplo, o caso em que o tempo de execução de uma tarefa depende da máquina na qual ela é executada; o caso em que as máquinas são idênticas; e o caso com uma única máquina. As variações também aparecem em restrições sobre as tarefas, como impor uma ordem de precedência, exigir execução atômica ou permitir subdivisão, impor deadlines para finalização e, então, não aceitar ou penalizar atrasos. Em todas elas, assume-se que não há necessidade de priorizar a execução de algumas tarefas em detrimento de outras. Já em cenários de criticidades variadas, tarefas mais e menos críticas competem pelo uso de um mesmo recurso. Essa necessidade surge em certos cenários industriais [Vestal 2007, Socci et al. 2013, Baruah et al. 2010]. Para ilustrar, citamos o compartilhamento da largura de banda em sistemas de comunicação [Hanzalek and Pacha 1998, Hanzálek et al. 2016]. Nos casos de criticidades variadas, cada tarefa está associada a um tempo estimado de execução e a um tempo máximo de execução, geralmente proporcional ao seu valor de criticidade. Em um escalonamento, permite-se que o tempo alocado a uma tarefa seja estendido até seu tempo máximo de execução. Assim, tarefas mais críticas podem receber mais tempo de execução para garantir sua finalização, causando, porém, o cancelamento de tarefas menos críticas. Para uma revisão recente sobre problemas de escalonamento em sistemas de criticidades variadas, indicamos [Burns and Davis 2015].

Consideramos o problema de escalonamento de tarefas com diferentes valores de criticidade em múltiplas máquinas de capacidade fixa. O objetivo é escalonar todas as tarefas usando o menor número de máquinas. Dado um conjunto $P = \{1, \ldots, n\}$ de tarefas, seja c_i o valor de criticidade da tarefa i. *A priori*, o tempo de execução esperado de cada tarefa é de uma unidade de tempo. No entanto, se uma tarefa i precisar de mais tempo durante a execução, seu tempo de execução pode chegar a, no máximo, c_i unidades de tempo, possivelmente causando o cancelamento de outras tarefas. É crucial garantir que, ao estender o tempo de execução de uma tarefa, nenhuma tarefa de criticidade maior seja cancelada. Além disso, consideramos um cenário em que todas as tarefas são, de certa forma, altamente críticas; para nós, isso significa que a criticidade de cada tarefa é pelo menos um fator da capacidade das máquinas. O objetivo é minimizar o número de máquinas necessárias para executar todas as tarefas.

Apresentamos um APTAS para o problema. Um APTAS para um problema de minimização é um algoritmo que, dada uma constante $\varepsilon \in (0, 1)$, produz uma solução cujo valor é no máximo $(1 + \varepsilon)\mathrm{OPT} + c$, sendo OPT o valor de uma solução ótima e c uma constante.

2. Um APTAS para Escalonamento de Tarefas Altamente Críticas

Denotamos $[n] = \{1, \ldots, n\}$. Dada uma constante $\varepsilon \in (0, 1)$, definimos um conjunto de instâncias I_ε para o *problema de escalonamento de tarefas altamente críticas* – ETC, sendo cada instância uma dupla (P, M), em que $M \in \mathbb{Z}_+$ é a *capacidade* das máquinas e $P = [n]$ é um conjunto de tarefas, cada uma com *criticidade* $c_i \in \mathbb{Z}_+$ tal que $M \geq c_i \geq \varepsilon M$, para $i \in [n]$. Formalmente, um *escalonamento* de um subconjunto de tarefas $P' \subseteq P$ é dado por um vetor de *instantes de início* $S(P') = (s_i)_{i \in P'}$ tal que quaisquer duas tarefas $i, j \in P'$, com $i \neq j$, satisfaçam $|s_i - s_j| \geq \min(c_i, c_j)$. A seguir, mostramos que essa definição de escalonamento é suficiente para garantir que nenhuma tarefa pode cancelar outra de criticidade maior. Dadas tarefas $i, j \in P'$, dizemos que i *compromete* j em $S(P')$ se $s_j \in [s_i, s_i + c_i)$, ou seja, a extensão da execução de i pode cancelar j.

Proposição 1. *Dados um subconjunto de tarefas $P' \subseteq P$ e um escalonamento $S(P')$, não existem tarefas $i, j \in P'$ tais que i compromete j e $c_i < c_j$.*

No nosso problema, o tempo máximo de execução de cada tarefa é definido por sua criticidade. Assim, dado um escalonamento $S(P')$ de um subconjunto $P' \subseteq P$, o tempo total de execução de todas as tarefas, chamado *makespan* de $S(P')$, é definido por $\max_{i \in P'} s_i + c_i$. Finalmente, definimos uma solução para uma instância (P, M) como uma partição $\mathcal{P} = P_1, \ldots, P_k$ de P e, para cada $j \in [k]$, um escalonamento $S(P_j)$ tal que $\max_{i \in P_j} s_i + c_i \leq M$. O objetivo é minimizar k. Definimos o valor de uma solução $\mathcal{P}$ para P como o número de máquinas utilizadas, ou seja, a cardinalidade de $\mathcal{P}$, e denotamos $k(\mathcal{P}) = |\mathcal{P}|$. Além disso, denotamos por $\mathrm{OPT}(P)$ o valor de uma solução ótima para P.

Para obter um APTAS, primeiro resolvemos uma versão mais restrita do problema, em que o número de criticidades diferentes é limitado por uma constante. Nesse caso, conseguimos obter uma solução ótima.

Lema 2. *Sejam $\varepsilon \in (0, 1)$ uma constante e $(P, M) \in I_\varepsilon$ uma instância do ETC com $|\{c_i : i \in P\}| = O(1)$. Em tempo polinomial, obtemos uma solução $\mathcal{P}$ com $k(\mathcal{P}) = \mathrm{OPT}(P)$.*

Demonstração. Como $c_i \geq \varepsilon M$ para toda tarefa $i \in P$, o número máximo de tarefas que cabem em uma máquina é $\eta = \lfloor M/(\varepsilon M) \rfloor \leq 1/\varepsilon$. Denote $|\{c_i : i \in P\}| = K$.

Chamamos de *configuração* uma tupla $F = (f_1, \ldots, f_K)$ tal que $\sum_{j\in[K]} f_j \leq \eta$, sendo f_j o número de tarefas com o j-ésimo valor de criticidade. Dizemos que um subconjunto $P' \subseteq P$ é *F-compatível* se o número de tarefas com o j-ésimo valor de criticidade em P' é igual a f_j, para todo $j \in [K]$. Dizemos que F é *viável* se existe escalonamento de um subconjunto F-compatível cujo makespan é no máximo M. Para verificar a viabilidade de F, selecionamos um subconjunto F-compatível P' e, para cada permutação de P', construímos um escalonamento em que a i-ésima tarefa na permutação é também a i-ésima tarefa escalonada, contando da esquerda para a direita. Se, para alguma permutação, o makespan é no máximo M, então F é uma configuração viável e retornamos o escalonamento obtido; caso contrário, F é inviável. Com contas simples, conseguimos construir o escalonamento referente a uma permutação em tempo linear em η e, para cada configuração, o número de permutações é no máximo $\eta!$. Como o número de configurações é no máximo $(\eta + 1)^K$, que é constante, obtemos o seguinte resultado.

Proposição 3. *Em tempo constante, obtemos todas as configurações viáveis e seus respectivos escalonamentos.*

Cada configuração viável representa uma maneira de escalonar um subconjunto de tarefas em uma única máquina. Assim, queremos escolher um conjunto mínimo de configurações viáveis e disjuntas que cobrem todas as tarefas. Para isso, usamos um programa linear. Seja $\mathcal{F}$ o conjunto de configurações viáveis. Criamos um conjunto de variáveis $\{x_F\}_{F\in\mathcal{F}}$, em que $x_F \in \mathbb{Z}_+$ indica quantas vezes a configuração F é utilizada na solução. Denotamos por n_j o número de tarefas com o j-ésimo valor de criticidade, $j \in [K]$. A função objetivo é minimizar o número de configurações utilizadas. Segue a formulação.

$$\begin{array}{lll} \text{minimize} & \sum_{F\in\mathcal{F}} x_F & \\ \text{subject to} & \sum_{F\in\mathcal{F}} f_j x_F = n_j, & 1 \leq j \leq K \\ & x_F \in \mathbb{Z}_+ & F \in \mathcal{F} \end{array} \tag{1}$$

Para cada $F \in \mathcal{F}$ tal que $x_F \geq 1$, construímos x_F subconjuntos F-compatíveis disjuntos. Seja OPT a coleção de tais subconjuntos. Retornamos a partição $\{P'\}_{P'\in\mathrm{OPT}}$ e seus respectivos escalonamentos. O programa linear garante que $|\mathrm{OPT}|$ é mínimo. O número de variáveis do PL é o número de configurações viáveis, que é limitado pela constante $(\eta+1)^K$. Nesse caso, o PL pode ser resolvido em tempo polinomial usando programação inteira de dimensão fixa, como em [Lenstra 1983, Eisenbrand 2003]. □

Agora, apresentamos um algoritmo que produz soluções quase ótimas para a versão geral do problema, isto é, quando o número de criticidades não é constante. Primeiro, particionamos o conjunto de tarefas em grupos de modo que tarefas de um mesmo grupo tenham criticidades parecidas e cada grupo tenha poucas tarefas. Para cada grupo exceto o primeiro, arredondamos a criticidade das tarefas para a maior criticidade dentre as tarefas daquele grupo. A partir disso, criamos uma *instância arredondada*, em que o número de criticidades diferentes é limitado pelo número de grupos e, portanto, por uma constante. Logo, usamos o algoritmo do Lema 2 para obter uma solução ótima para a instância arredondada, a partir da qual construímos uma solução viável para as tarefas originais arredondadas, isto é, todas exceto as do primeiro grupo. Como o número de tarefas em cada grupo é pequeno, conseguimos adicionar as tarefas do primeiro grupo ao escalonamento já obtido sem aumentar muito o número de máquinas. Para isso,

usamos a técnica de arredondamento conhecida como *linear grouping*, introduzida por [de la Vega and Lueker 1981] para o problema de *bin packing* unidimensional. No próximo teorema, mostramos um APTAS para o problema ETC.

Teorema 4. *Dada uma constante $\varepsilon \in (0, 1)$, existe um algoritmo de tempo polinomial que, para qualquer instância $(P, M) \in I_\varepsilon$ do ETC, produz uma solução $\mathcal{P}$ para P tal que $k(\mathcal{P}) \leq (1 + \varepsilon)\mathrm{OPT}(P) + 1$.*

Demonstração. Defina $\delta = \lceil \varepsilon^2 n \rceil$, em que $n = |P|$. Ordene as tarefas em ordem não crescente de criticidade. Nessa ordem, forme $m+1$ grupos $G_0, \ldots, G_m$ tais que $|G_j| = \delta$ para $j \in \{0, \ldots, m-1\}$ e $|G_m| \leq \delta$. Vamos, por ora, esquecer G_0 e arredondar as demais tarefas. Para $j \in [m]$, criamos um grupo $\overline{G}_j$ a partir de G_j arredondando a criticidade de cada tarefa em G_j para a maior criticidade dentre todas as tarefas em G_j. Denotamos $L' = \cup_{j \in [m]} \overline{G}_j$. Nosso objetivo é, a partir de uma solução ótima para L', construir uma solução quase ótima para P. Para isso, primeiro mostramos que uma solução ótima para L' não usa mais do que $\mathrm{OPT}(P)$ máquinas. Note que, pela ordenação das tarefas antes de particioná-las em grupos, toda tarefa em G_j tem criticidade maior ou igual à criticidade das tarefas em $\overline{G}_{j+1}$. Assim, dada uma solução ótima para P, construímos uma solução para L' da seguinte forma. Para $j \in \{0, \ldots, m-1\}$ substituímos cada tarefa do grupo G_j por uma tarefa do grupo $\overline{G}_{j+1}$ e descartamos as tarefas do grupo G_m. Isso dá uma solução para L' usando $\mathrm{OPT}(P)$ máquinas, implicando que $\mathrm{OPT}(L') \leq \mathrm{OPT}(P)$.

Agora, vamos de fato obter uma solução para L'. Como o número de criticidades diferentes em L' é no máximo $m \leq n/\delta \leq 1/\varepsilon^2$, usamos o algoritmo do Lema 2 para obter uma solução ótima para L', a partir da qual construímos uma solução para $P \setminus G_0$ da seguinte forma. Note que toda tarefa em G_j tem criticidade menor ou igual à das tarefas em $\overline{G}_j$. Assim, substituímos cada tarefa arredondada por sua correspondente original, obtendo uma solução para $P \setminus G_0$ usando $\mathrm{OPT}(L')$ máquinas. Como a instância tem n tarefas de criticidade pelo menos εM e as máquinas têm capacidade M, o valor $\lceil n \cdot \varepsilon M / M \rceil$ é um limitante inferior para o valor ótimo de P, ou seja, $\mathrm{OPT}(P) \geq \varepsilon n$. Assim, para adicionar as tarefas de G_0 à solução obtida para $P \setminus G_0$ sem aumentar muito o número de máquinas, basta alocar cada tarefa em G_0 a uma nova máquina. Obtemos, então, uma solução para P usando $\mathrm{OPT}(L') + |G_0| \leq \mathrm{OPT}(P) + \lceil \varepsilon^2 n \rceil \leq \mathrm{OPT}(P) + \varepsilon \cdot \varepsilon n + 1 \leq (1 + \varepsilon)\mathrm{OPT}(P) + 1$ máquinas. □

3. Considerações Finais

Apresentamos um APTAS para o problema de escalonar tarefas de criticidade variadas em múltiplas máquinas. Na versão abordada, o tempo adicional permitido a cada tarefa é igual ao seu valor de criticidade. Acreditamos que, com poucas modificações, nosso algoritmo funciona para uma versão mais geral do problema, em que o valor de criticidade e o tempo máximo de execução de cada tarefa não são necessariamente iguais. Isso faz sentido quando há tarefas muito críticas para as quais o tempo estimado de execução não é tão alto. Pode ser um desperdício reservar um tempo extra tão maior que o necessário para tais tarefas, arriscando assim o cancelamento de um número maior de tarefas menos críticas. Como trabalhos futuros, propomos investigar versões com custo nas tarefas, e também a versão em que não há limitante inferior nos valores de criticidade.

Esse projeto foi parcialmente financiado por CNPq (Proc. 313146/2022-5, 404315/2023-2, 404779/2025-5) e FAPESP (Proc. 2022/05803-3).

Referências

Baruah, S., Li, H., and Stougie, L. (2010). Towards the design of certifiable mixed-criticality systems. In *2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 13–22.

Burns, A. and Davis, R. I. (2015). Mixed criticality systems - a review.

de la Vega, W. F. and Lueker, G. S. (1981). Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1:349–355.

Eisenbrand, F. (2003). Fast integer programming in fixed dimension. In Di Battista, G. and Zwick, U., editors, *Algorithms - ESA 2003*, pages 196–207, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hanzalek, Z. and Pacha, T. (1998). Use of the fieldbus systems in academic setting. In *Proceedings Real-Time Systems Education III*, page 15.

Hanzálek, Z., Tunys, T., and ůcha, P. (2016). An analysis of the non-preemptive mixed-criticality match-up scheduling problem. *Journal of Scheduling*, 19:601 – 607.

Lenstra, H. W. (1983). Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548.

Socci, D., Poplavko, P., Bensalem, S., and Bozga, M. (2013). Mixed critical earliest deadline first. In *2013 25th Euromicro Conference on Real-Time Systems*, pages 93–102.

Vestal, S. (2007). Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium*, RTSS '07, page 239–243, USA. IEEE Computer Society.