國立臺灣大學管理學院資訊管理學系

碩士論文

Department of Information Management

College of Management

National Taiwan University

Master Thesis

考慮公平性、不同機臺產能與不同工作截止時間的工作分配問題

A Job Allocation Problem Considering Fairness, Various Machine Capacity, and Various Job Due Dates

張鵬路

Peng-Lu Zhang

指導教授：孔令傑 博士

Adviser: Ling-Chieh Kung, Ph.D.

中華民國 107 年 7 月

July, 2018

doi:10.6342/NTU201801942

致謝

當我剛剛來到台大，開始一個全新的領域的學習的時候，心中充滿著對自己以及對專業的懷疑。這個專業真正要學什麼，我自己又是否能夠完成學業。每一天都是全新的領域，全新的知識。在魏老師的研究組，學到的是文字探勘以及舊藥新用方面的知識，缺乏程式經驗讓我很難繼續下去。魏老師並沒有放棄我，反而推薦我來到孔老師的研究組，開始接觸資訊經濟、工作排程、修建設施的最佳地點選擇這些全新的方向。在課堂中，管理資訊系統、資訊創新、資料探勘、雲計算、資訊經濟、知識管理，這些課程雖然都給我帶來了很多的幫助，但是也讓我無所適從，我一度懷疑我究竟能否找到一個方向來把這篇論文完成。實在是感謝孔老師能夠有足夠的耐心，幫我安排課程的修行方案，不厭其煩的在我沒有進度的時候耐心的和我進行一對一的交流。哪怕每次的進度很小，哪怕有不少的點子甚至是在和老師討論前走往會議室的路上靈機一動的，孔老師耐心的等了我額外的兩年，我也終於把這些一點點的小點子匯集起來，完成了這篇文章。這篇文章仍然留存著不少的缺點，從算法設計的可能會把一個比較差又足夠大的工作沉在序列的前方，到具體的參數試驗只用了少量的案例，這些缺陷，如果我能早早的利用好時間，可能就能有所彌補了。到了這個時候，卻又留下了一絲念想，如果這個方法能夠讓孔老師繼續研究，讓學弟學妹們在這個基礎上又解決一些更為現實的問題，例如加入工作的起始時間？如果我能夠更努力一些，或者說像最後一學期那樣努力的話，可能就能寫入這篇文章裡了吧。從騏瑋學長的文章裡獲得靈感，如今自己要離開了卻又留下一個「坑」給孔老師以及學弟學妹們。家裡也義無反顧的支持一度沉淪的我。真的是靠各位扶著才能完成這個演算法呢。這四年來，實在是給各位，給孔老師，給魏老師，給父母，給同學，給舍友添麻煩了。

張鵬路

107 年 7 月 23 日

doi:10.6342/NTU201801942

摘要

工作分配在各領域中一直是一個重要議題。在製造業中，其中一個目標是希望能最大化最小利潤的機台。然而，在實際的工作分配中，我們不僅僅要考慮工作的利潤，還要考慮工作本身是否可以被完成。工作的截止時間和機器的產能限制都是安排工作時必須考慮的地方。這樣的特性，讓工作分配更為困難，乃至於我們需要一個專門的排序法來解決這個問題。我們希望能設計演算法，在考慮公平性、不同機臺產能與不同工作截止時間的工作分配問題，有效分配工作。

在本研究中，我們考慮若干工作及若干機台，完成工作會帶來利潤，但也會為機台帶來工作量，各機台能負荷的最大工作量是一樣但是有不同的情形，同時每個工作有自己的截止時間。為了使各機台的利潤盡量地一致，我們的目標是最大化得到最小利潤的機台上的利潤，根據前人的最長處理時間優先演算法（LPT rule），以及產能限制下的最大收益優先演算法（CHBF rule），我們提出了最差收益擠出法。

我們採用數值試驗的方式證明了最差收益擠出法的有效性。通過和整數規劃得出的最優解以及線性規劃得出的最優解的上界進行比較，我們可以顯示最差收益擠出法是一個優秀的算法。通過和基因（GA）算法的比較，我們證明了我們的算法在耗時比基因算法少的情況下有著與之相當甚至更為優秀的表現。我們認為這個算法在未來還有一些問題可以應用，本文所述的「擠出」概念以及具體的擠出方法可以為今後的研究者們提供一種解決問題的新思路。

關鍵字：工作分配，公平性，截止時間，近似算法，擠出算法。

doi:10.6342/NTU201801942

Abstract

Job scheduling is an important issue applied in many fields. In the manufacturing industry, one of the objectives is to assign jobs to machines in order to maximize the minimum profit among machines. Assigning jobs by considering their benefits is a good idea. However, we also have to consider whether the jobs can be assigned to those machines, because each machine cannot be assigned too many jobs due to limited capacity. Moreover, each job has its own due date. This characteristic introduces a new challenge to this allocation problem and calls for a specific algorithm to solve it. We propose an efficient algorithm to assign jobs by taking fairness, machine capacity, and job due dates into consideration.

In this study, our objective is to assign jobs to bring benefits to all the machines as equally as possible while ensuring that machines cannot be overloaded and all assigned jobs can be completed before their due dates. The capacity of all machines are the same. We propose the least benefit out (LBO) algorithm based on the longest processing time (LPT) rule and the capacitated highest-benefit job first (CHBF) algorithm.

We use numerical experiments to verify the efficiency of the LBO algorithm. By comparing with optimal solutions or their upper bounds obtained by integer/linear programing, we confirm that our algorithm can get a good enough result. Moreover, we choose Genetic Algorithm as a benchmark and find that LBO can use less time to get similar or better results in almost all cases in our experiment. We believe that this algorithm as well as the way to squeeze out worse works can be a new groundbreaking method that may be adopted for solving other problems.

Keywords: job scheduling, fairness, due date, heuristic algorithms, squeezing out.

doi:10.6342/NTU201801942

目錄

doi:10.6342/NTU201801942

圖表目錄

doi:10.6342/NTU201801942

第一章 緒論

第一節 研究動機

作為一個經典的最佳化問題，平行機台工作分配問題已經有了數十年的研究歷史，隨著科學的發展，類似的問題也出現在各行各業的研究中（Pinedo，2012）。不僅僅是傳統的工業工程領域，隨著計算機科學的發展，類似的平行機台問題在計算機領域中也得到了應用。使得這一問題引起了更為廣泛的興趣。

然而在近些年的實踐與學術討論中，依然有不少新的問題出現。在生產實踐中，原始的平行機台問題更多的是專注於工作的盡早完成，這忽視了工作量與收益的取捨問題，當機台的負擔成為一個必要考慮的因素時，僅僅是盡早完成所有工作的排序手段已經無法得出合適的解。如果進一步考慮到每個工作有其不同的截止時間，那麼問題將進一步複雜。於此同時，如果將問題擴展到更多的領域，我們發現，比起傳統排序法中我們將各種問題分開，獨立排序的理想情況，現實中的情況更為複雜，而且與傳統排序法有所衝突。例如在工廠分配的任務中，工作的截止日期如果可以不同，則相比較只允許足夠久的截止日期的排程方案，工廠將可以接受更為靈活的訂單。這將可以在競爭中獲得優勢。尤其是在如今網路時代中，更為靈活的接單方案就代表了可以嘗試採用離線商務模式（O2O），接受更為多樣化和具體的方案。同時合理的排序方法也可以排除那些不合適的工作，起到有效的篩選作用。此時，我們就有必要在傳統的只考慮完成工作數或者最多延遲時間的排序法之上，結合實際工作收到的利益，得出一個可以解決這種更為複雜的訂單的排序法。

在此之上，我們也希望能夠將目標定位在公平排序而不僅僅是利益優先。一方面是因為在過往的研究中（Liu，2016），我們發現時間上的

doi:10.6342/NTU201801942

公平排序方案和最短時間排序方案是接近的方案。我們不需要過度擔心公平排序的結果造成糟糕的利益表現。更重要的是，相比較利益優先方案，公平排序擁有更為優秀的實踐價值。例如大型企業在分配工作時，如果旗下的各個代工廠拿到的工作收益不同，收益更高的代工廠商將會拒絕給收益較低的代工廠分享利潤。這樣一方面可能會傷害低收益廠商本身的運營，同時也會影響他們與企業的合作。因此我們希望能夠得到一個以公平為目的，可以滿足工作本身一定程度上多樣化限制這樣一個順應時代的排序方法。

第二節 研究目的

在這個研究中，我們將會考慮上文所述的工作排程問題。為了表述方便，我們將會以工作與機器這一情境來展示這類問題。其中，我們的研究最重要的創新在於試圖在傳統的平行機台問題上，用一個統一的排序法同時解決多種限制條件下的公平分配問題。當然，這些工作之間都是相互獨立且不可分割的。

首先我們將會討論在平行機台中，每台機器有不同總工作量限制的情況，並且希望我們的排序法能在工作量和收益沒有特定關聯的情況下依然獲得較好的表現。第二步則是研究的重點，我們希望允許每個工作有其自身的最晚完成時間。最後我們將探討算法進一步拓展的可能性，例如另一種公平的定義，認為收益應該和工作量限制呈一定比例，亦或者每台機器有著不同的效率。在前兩個階段，我們的目標式是最大化總利益最小的機台。在第三階段，將會根據具體情況進行調整。

根據 Liu（2016），我們研究的問題是 NP-hard 的問題。因此，我們將構建一個時間複雜度在多項式時間內的啟發性演算法（heuristic algorithm）。我們將基於兩大經典排序法 Longest Processing Time（LPT）

doi:10.6342/NTU201801942

和 Earliest Due Date（EDD），通過試圖尋找兩者結合的方法，實現一個可以滿足存在不同最晚完成時間工作的情況下，實現公平分配的方案。其中，LPT 方法已經在公平分配問題上得到許多成熟的運用（Csirik et al.，1992；Deuermeyer et al.，1982；Graham，1966；Graham，1969；Liu，2007），而 EDD 則是一個針對不同最晚完成時間的工作的經典排序方法。

第三節 研究架構

在下一章中，我們將對現有的研究結果進行一個綜述。在第三章中，我們將用數學模型定義我們的問題。第四章我們將詳細介紹我們的算法。第五章中，我們將用數學模擬的方式來驗證我們的算法的表現。

doi:10.6342/NTU201801942

第二章 文獻探討

第一節 平行機台的工作排序

平行機台的工作排序是工作排序問題中一個重要的部分。(Pinedo，2012)的著作中將工作排序問題分為兩個基本大類，一是已知所有的工作和機器情況，一是工作的情況暫時未知。對於我們希望討論的工作和機器情況確定的問題，主要研究的是兩大問題，一是總完成時間問題，另一是工作有期限情況下的問題。這兩個問題各自有過相對廣泛的討論。

總完成時間問題最早由 Graham（1966、1969）的兩篇文章證明了 LPT 算法的邊界，使得 LPT 算法，即最長的工作先排成為了這類問題的主流解法。方法是通過構建一個特殊的情形，並且證明 LPT 排序法排出的結果不會比這種情況差。從這兩篇文章起，LPT 方法一直是該問題的核心解決方案。在 1976 年，Coffman（1976）針對幾種 LPT 已經其衍生算法進行了總結，並且比較了幾種方法的優劣。這盤文章雖然沒有撼動 LPT 算法的地位，但是也拓展了這類算法的可能以及可能面臨的問題。例如 SPT 算法雖然看似和 LPT 相反的算法，實際上當進入排序時，SPT 算法和 LPT 并不是完全相反的關係，在每台機器上都有工作后，SPT 的排序會讓工作排在最短的那台機器上，如果我們把每台機器上都被排過一次視為一輪，LPT 算法會讓下一輪中最短的工作排在上一輪后最長的機器那裡，而 SPT 則讓其排在最短的那台機器上。因此為了解決這一問題，對 SPT 算法進行一個改進是必須的，即排序前搖讓工作先分好組，然後進行排序。先分組的再進行排序也成為了研究新排序法的一種常用方法。Coffman 並沒有針對 SPT 排序法設計一個明確的排序，而是把滿足先分類，再按照分類的長短從短往常排的所有結果都視為 SPT 排序法的結果的一種。除了數學證明，Walter（2013）亦對

doi:10.6342/NTU201801942

LPT 進行了數值上的研究。

Deuermeyer et al.（1982）證明了 LPT 算法不僅在傳統的最快完成問題上非常有效，在為了公平的 max-min 問題上，亦非常有效。在 Csirik（1992）的文章中，這個最差表現被證明為趨於 4/3。於此同時，一些其他的算法也試圖從別的地方解決這類問題，Tsai（1992）提出了 RLD 算法以及其的進化版 DIF 算法。其思想在於，對於公平類問題來說，我們只需要考慮各個機器上目標式的差值，而不需要考慮工作是怎麼樣的，因此在排序前先依照機器數量分組關注每個分組之內的差值，并依照這個差值排序。而 DIF 算法中，我們甚至不需要考慮已經排好的結果是什麼樣的。這類算法在部分分佈的排序結果上比起 LPT 算法更有潛力，但是由於算法採取了排序前先分組的方案，導致了默認情況下最終解中各台機器上工作數量相同。反而限制了算法的表現。He（2003）則將排序前先分組的思路進一步推廣，他們直接依照工作長度進行分組，先限制每組之內的差值再進行排序。因為兩組之間排序后的結果必然小於兩組中差距較大的那一組，排序后的結果也不會比開始設計的分組值差。然而這類算法在進一步推進更為精確的最差情況的同時，也限制了排序法的應用可能。和 LPT 運用于多種情境下不同，這類算法後繼寥寥。但是這些工作也展示了分組排序法潛在的價值。Liu（2016）提出了 CHBF（機器工時限制下的最高利益優先）排序法作為 LPT 排序法的一個變種，證明了 LPT 類算法在考慮到利益的情況下依然有著可以利用的地方。在不考慮機器限制（Capacity）的情況下，CHBF 算法簡化后的 HBF 算法將是我們的基礎算法。

於此同時，對於 JSD 問題，即擁有最後截止日期的工作的排序問題。Lawler（1969）曾有過討論，建議的方法是把這個條件簡化為一部分工作都有個共同的截止日期。但是文中也提到這種設計下會造成巨大

doi:10.6342/NTU201801942

的計算負擔。由於在這之後的研究基本都專注于解決超過時間的時間總數的減少之類的題目，而不是超過時間的工作數量的減少，在這裡就不作贅述。這類研究趨於停滯可能是因為嘗試把工作可以安排的時間單單作為一個限制條件而不是作為要改善的總目標在排序法上有不易解決的內容。這也說明了在這個問題上需要一些創新。

第二節 公平問題和 max-min 問題

對於公平的界定，Bertsimas（2011）曾著文專門討比較過比例公平（PF）和 max-min 公平的特點。在這裡，我們希望使用 max-min 公平來代表本研究中的「公平」。因為比例公平使用在資源分配問題上時，其目的是最大化每個獲得者的收益的對數和。這在分配一個可以分割的資源時是合適的，但是在分配不可分割的工作時這會導致我們求最優解的過程過於複雜。而根據之前我們對於平行機台問題的討論，我們可以看到我們所求的最差情況相比比例公平求解要寬鬆很多。我們認為每個工廠的需求是相同的，一方面在問題設計上工廠產能的差別會符合其作為一個限制條件的特點，我們希望產能的差別這一條件是獨立的；另一方面從現實中考慮，時間問題帶來的需求不同是微小的，不同廠商除非發生變故，不太會出現開工時間巨大的差異，而生產力的需求差別則是一個更為複雜的問題，這對於當前問題是一個進階的問題。在這種情況下，相比算法帶來的總體情況的提升，例如從 EDD 算法或者 LPT 算法排序后刪除不滿足的工作這樣的算法到我們設計的算法，這之間的差距可能要遠大于採用 max-min 和比例公平的差別。因此採用 max-min 也是公平排序中較為主流的目標式。

Deuermeyer（1982）首次詳述了排序算法中的 max-min 問題， 他成功的證明了 max-min 問題也是一個 NP-hard 問題，同時採用 LPT 排

doi:10.6342/NTU201801942

序法時也擁有同樣的最差邊界 4/3。這也給予了我們以 LPT 算法為起點進行研究的信心。

doi:10.6342/NTU201801942

第三章 問題描述與建模

第一節 問題描述與建模

在這篇文章中，我們考慮這樣一個問題。我們要分配n個工作去m台機器上。其中工作所屬的集合為J即$n \in J$。 每個工作J有其利益b_j, 需要的時間c_j， 以及工作的截止時間D_j。機器的集合為I即$i \in I$.而每台機器i有其最大容量K_i。 我們將試圖最大化利益最小的機台。因此，我們需要一個參數來表示工作分配給哪台機器x_{ij}：

$$x_{ij} = \begin{cases} 1 & \text{如果工作} j \text{被分配到了機器} i \text{上} \\ 0 & \text{其他情況} \end{cases} \quad i \in I, j \in J$$

直觀上來看，我們不但要決定如何把工作指派給機器，還要決定在一台機器上工作的先後順序。但是，我們注意到 EDD 演算法在一台機器的情況下，是求最小化超過工作截止時間的工作數量的優化問題的最佳解(Pinedo,2012)。 由於我們的問題的解中，不允許有超過工作截止時間的工作的存在。 即超過工作截止時間的工作數量為 0。 那麼如果我們對我們的一個可行的解進行處理，將其在每台機器上的排序次序都採用 EDD 演算法排序，那麼這樣處理下來的結果，依然滿足超過工作截止時間的工作數量為 0，即這依然是一個符合我們要求的解。 同時，由於工作的先後順序並不影響我們這個議題下收益的計算，所以如果我們已經決定好了這個工作該分配給哪台機器，那麼將其工作順序調整為 EDD 排序也不會影響我們的結果。 因此，所有的適用解都有其對應的 EDD 形式，我們只需要針對性的研究其 EDD 形式的解即可知道這個解的好壞。 總而言之，我們只需要考量在每台機器上各自實行 EDD 排序後的工作分配結果，就可以找到最佳的工作分配方式。 於是，我們便不需要額外考慮工作在機器上的先後順序，只用考慮工作分配本身即可。

doi:10.6342/NTU201801942

在不失一般性的前提下，我們假設給定的工作已經按照截止時間排序了，亦即

$$D_1 \leq D_2 \leq \cdots D_n。$$

我們定義決策變數 w_{ij} 為當前工作x_{ij}的開始時間。利用這個中間變量，我們就確定了一個工作在當前排程結果下所排的位置。x_{ij}決定了工作被分配給的機器，而w_{ij}決定了工作在機器上被排的位置。

$$w_{ij} = R，\ 工作j在機器i上的起始時間\ i \in I, j \in J$$

在$x_{ij} = 0$時，即工作沒有被排入機器i，w_{ij}理論上應該不存在，我們會在後文中解釋這個問題。

根據 x_{ij} 和 w_{ij}，我們的最佳化問題可以被寫成下面的整數規劃模型：

$$\begin{aligned}
\max \quad & \min_{i \in I} \left\{ \sum_{j \in J} b_j x_{ij} \right\} & \\
\text{s.t.} \quad & \sum_{i \in I} x_{ij} \leq 1 & \forall j \in J \\
& \sum_{j \in J} x_{ij} c_j \leq K_i & \forall i \in I \\
& w_{ij} \geq 0 & i \in I, j \in J \\
& w_{ij} \geq w_{i,j-1} + c_j \cdot x_{i,j-1} & \forall i \in I, j \in J, j > 1 \\
& w_{ij} \leq D_j - c_j \cdot x_{ij} & \forall i \in I, j \in J
\end{aligned}$$

於是， k 從式子中被排除出來。注意到此時第五條限制式使得所有的w_{ij}都受到了D_j的限制，這看上去是出現了額外的限制式使得不落在機器i上的工作也受到了與機器i相關的限制。但是我們可以證明這個限制式不會影響結果：

已知，假如工作 1 在機器i上，由於機器上只有這一個工作，而工作本身的$c_j \leq D_j$，第五條限制式是肯定成立的。

同時，如果工作j_0滿足第五條限制式，對於任何工作$j > j_0$，我們已

doi:10.6342/NTU201801942

知工作在 EDD 排序下，有$D_j \geq D_{j_0}$，所以如果工作j和工作j_0不在同一個機器上，則工作j也是滿足限制式的。因此如果要使得工作j不滿足限制式，就必須找到一個相對的j_0違反限制式。而j_0違反限制式則意味著這不是可行解。因此得證任何不是排在機器第一個的工作，都不會受到額外的影響。因此，「多餘」的w_{ij}並不會在限制式中對工作沒有被排入的機台產生額外的影響。

雖然利用 EDD 排序的特性在實際的算法中並沒有直接的體現，但是得證 EDD 的最優性在我們的排序法中給予了我們一個優化的目標。即如果我們讓我們的算法無限次運行下去之後，我們的算法應該讓每個機器上被分配的工作趨近 EDD 排序。此外，在計算整數規劃的最佳解以及利用線性規劃來取得結果的下界時，利用 EDD 排序也可以帶來極大的方便。

最後我們可以得到以下參數表：

參數	
I	機器的集合
J	工作的集合
n	工作的數量
m	機器的數量
L_j	當前機器上比當前工作早完成的工作以及該工作本身的集合
K_i	機器的工作時間限制 $(K_i > 0)$
c_j	工作的時間長度 $(c_j > 0)$
b_j	工作的利益 $(b > 0)$
D_j	工作的最晚完成時間 $(D_j > 0)$

*決策變量見下頁

doi:10.6342/NTU201801942

決策變量	
$x_{ij} = \begin{cases} 1 & 如果工作j被分配到了機器i上 \\ 0 & 其他情況 \end{cases}$	$i \in I, j \in J$
$w_{ij} = R$，工作j在機器i上的起始時間	$i \in I, j \in J$

表格 三.1 符號表

第二節 NP-hard 問題

根據 Liu (2016) 的研究，在不考慮工作含有最晚完成時間時，此問題已經是 NP-hard 問題，而加入工作含有的最晚完成時間後，其不含最晚完成時間就是這個問題視為無窮大最晚完成時間的簡化，所以也是 NP−hard 問題。此外，對於這個問題而言，其特殊性在於當選擇了一個工作分配的結果時，例如機器 1 獲得編號為 1 3 4 5 的工作但是這個解無法滿足不超過工作截止時間或者機器容量的要求，需要選擇「最合適」的工作時，這個問題本身就是一個 NP−hard 問題 partition 的變種，因此在設計演算法時，我們無法加入分配集合X的工作到機器Y， 選擇集合X中最合適的子集x這樣的步驟，而需要一個具體的尋找子集的手段。

doi:10.6342/NTU201801942

第四章 算法介绍

第一節 最差收益挤出算法简介

我們希望能夠獲得一個多項式時間內的排序方法，使得這個排序法的結果可以較好的解決考慮公平性、不同機臺產能與不同工作截止時間的工作分配問題。使其成為一個可以在生產實踐中直接廣泛使用的啟發性演算法。這個演算法在工作截止時間限制比較弱時，能夠接近傳統的平均分配演算法，而在工作截止時間限制較強時，能夠接近 EDD 演算法。 但是，傳統的工作平均分配演算法和 EDD 演算法差異巨大，直接使用貪婪演算法來量化每個工作的「適合度」並不容易，因此這裡我們反其道而行之，提出了最差收益擠出(Least Benefit Out)演算法。

在傳統的排序法中，我們不會將已經決定好排序的工作再從排好的佇列中拿出。 但是在這個問題中，我們發現，由於 EDD 演算法是一個完全按照工作截止時間的排序演算法，而傳統公平分配排序法 LPT 或者 RLD 都是根據工作本身的時長來進行排序的。 這兩者是完全不同的維度。 對於有機器最大工作量限制的公平利益問題，Liu(2016)在研究中發現 CHBF 即工作量限制下的利益優先排序法表現優秀。 該演算法則是將 LPT 中的工作時長替換為利益，這依然不能改變其和 EDD 演算法的主要參考參數完全不同維度的事實。 但是無論 LPT 排序法或者是 RLD 排序法亦或者是 CHBF 排序法，這三者的共同特點就是將對結果影響最少的元素放到最後去排序。 即最短的工作最後排，工作時間差最短的組合最後排，利益最小的工作最後排。 受此啟發，只要我們將這些最「差」的工作放在演算法考慮中的最低優先順序，那麼就可能得到一個表現還不錯的演算法。 對於我們這個有著機器最大工作量已經工作完成時間這兩項限制的問題而言，直接尋找最合適的工作不容易，

doi:10.6342/NTU201801942

但是如果是在排序的過程中，尋找最差的工作卻相對容易，違反上面兩條限制式的工作組合，毫無疑問就有潛在的差工作， 而在這些工作中收益較小的工作則可以認為更有可能是一個合適的擠出物件，先進行傳統排序，再根據限制式把較差的工作擠出，便是最差工作擠出演算法(LBO)解決此類問題的基本思路。

具體到演算法上，我們採用了一些取巧的方法來尋找出合適的擠出物件。 但是從實驗結果看來，這一演算法對於這類問題是普遍有效的。

第二節 最差收益挤出算法具体步骤

最差收益擠出算法總體來說分為三個部分，其一是分配部分，其二是擠出部分，其三是循環部分。一般來說，傳統演算法只會有分配部分，不會出現工作擠出的情況，故具有其他兩個部分這也是最差收益擠出法與傳統排序法本質的不同。雖然在一些啟發式算法中可以出現目標式暫時下降的改變，例如基因演算法里允許突變或者交叉出一個比目前最佳解稍差的結果，但是這些算法的「擠出」部分是隨機的而沒有目的性的，雖然結果較不錯，但是隨著工作數量增加，表現下滑比較嚴重。相比之下，以擠出為重要步驟的 LBO 算法在工作數量較多時依然可以保持較好的結果以及效率。

分配部分是決定工作分配給哪台機器的部分，即按照一定的規律，把工作分配給各個機器。對於公平分配問題來說，常用的有 LPT，SPT 等方法，Coffman（1976）曾經對此進行了比較。但是對於我們這個問題來說，由於允許工作在排入后再被從已經排好的工作序列中擠出，所以這一部分並不是本算法的優化重點。

擠出部分是根據限制式來把工作擠出的部分。 工作以一定順序排入機器後，如果遇到不符合限制式的情況，將不得不把這些不符合限制

doi:10.6342/NTU201801942

式的工作擠出。工作擠出這一問題本身也是 partition 問題的變種，當工作的收益等於工作時長且最晚完成時間為總工作時長合的一半時，在一台機器上的選擇被擠出的工作的問題就是一個典型的 partition 問題。擠出部分實質上就是一個利用擠出規則來選出應該留在這台機器上的工作的過程。擠出后的工作會在循環部分進一步判斷是否要繼續參與排程，暫時不參與排程的部分會暫時被歸入棄用池O，工作並不會被永久棄用直到循環到最後一輪。

循環部分是決定被擠出的工作是否繼續參與排序的部分。如果我們直接把不符合限制式的工作拋棄，因為限制式的複雜性，必然會有大量可以排進去的工作沒有成功的排入。適當的循環可以保證算法在快速完成的同時將工作盡可能的排入。循環部分中，我們將工作曾經被排入的次數統計起來，作為工作是否「好排」的標準。如果一個工作曾經在循環部分被排入排出了 3 次，我們便用$g=3$來表示其排入的次數。調整合適的g值上限可以影響算法的表現，但是根據實驗，目前看來選擇g_{max}=2 即可以做到很好的表現。同時，循環次數較小的工作會被優先進行排序嘗試。在實作中，利用循環次數取負即$-g<0$來標示被暫時歸入棄用池的工作O。

工作池P是目前還需要排的工作的集合。工作池的工作的來源有以下兩種：在工作池分配部分從總工作集合J中放入的工作；在擠出部分從機器上返還的工作。而工作池中的工作也有兩個方法流出：在分配部分排入機器中的工作;在循環過程中被判定不適合繼續參與排序的工作。工作池的目的是控制當前所排的工作的數量。最差收益擠出法的總體流程如下圖所示：

doi:10.6342/NTU201801942

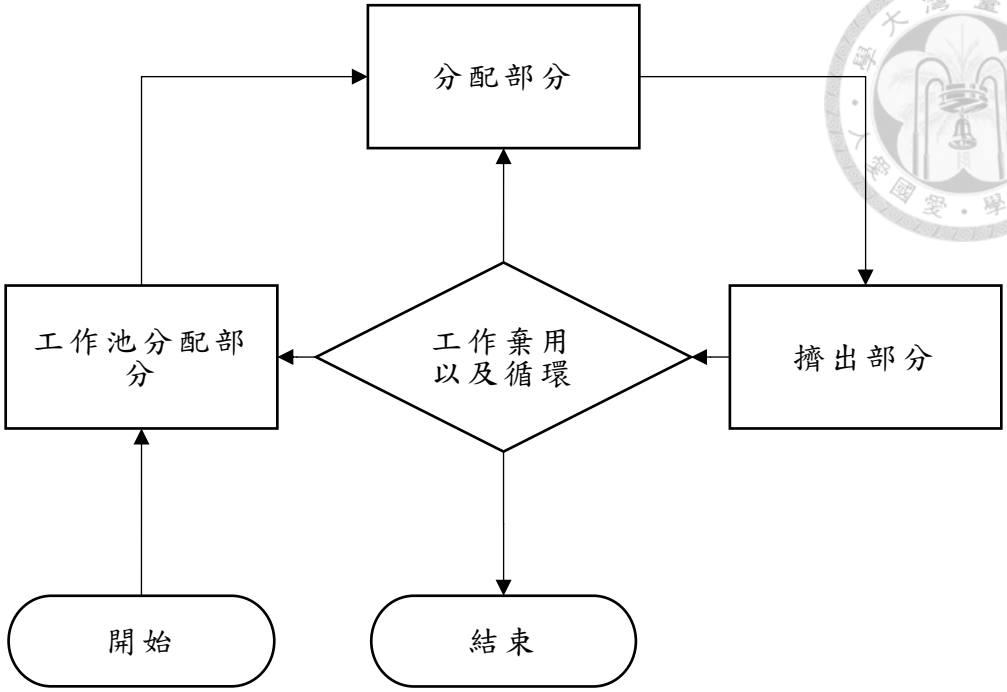

图 四.1 演算法基本流程圖

排程相關新增參數表	
P	工作池中的工作的集合
O	棄用池中的工作的集合
R	排序結果的集合
g_j	工作的循環次數

表格 四.1 排程相關新增參數表

1 初始化部分

對工作進行適當的補足以滿足新的工作數量n是機器數量m的整數倍例如工作如果數量為 15，機器為 4 臺，則補足額外的 1 個工作。在實際運算中，一般採用$c_j = 0$，$b_j = 0$，$D_j = M$ ，進行補全，並且會額外多補一組來讓需要排的原始工作進行一次額外的迴圈過程。

2 工作池分配部分

2.1 取出m個工作，放入工作池。

doi:10.6342/NTU201801942

2.2 初始化這些工作的循環次數$g_j = 0$。

3 分配部分

3.1 將工作池中的工作以利益從大到小排列，即$b_p \geq b_{p+1}$，$p \in P$。

3.2 查看工作的循環次數即g_j值以及當前循環$g_{current}$。初始化$g_{current} = 0$。

3.3 跳過$g < 0$的工作，并把屬於當前循環的工作$g_j = g_{current}$取出，將其中排序最靠前的工作排入當前總利益最小的機器。如果沒有，則$g_{current} = g_{current} + 1.$

3.4 在這一階段，我們以取出一個工作j_0為例。值得注意的是，分配階段每次運作是分配一個工作給某台機器。而這個分配規則本身可以是我們現在使用的 LPT 的變種 HBF，也可以是 RLD，EDD，SPT 等算法其本身或者變種。

4 擠出部分

4.1 查看取出的工作j_0的循環次數g值并使$g_j = g_j + 1$。并將j_0排入一個較為合適的位置。具體規則如下：把j_0放在要排到的機器的第一個工作。然後執行兩個判斷。

4.1.1 j_0是否違反了他自己的限制式要求。

4.1.2 j_0之前的工作如果存在，是否比j_0的收益更小。

兩者如果都不違反，則將j_0往後移 1 位繼續判斷。如果違反，則把j_0排在不違法這兩個要求的最後一個位置。在這個操作后，之後的工作會被擠到後面的位置，也有可能會違反其相關的限制式而被「擠出」，這也是這個算法得名的原因。

4.2 從機器的首個工作開始檢查排序的結果，如果出現違反工作完成時間等限制的情況，則將違反限制的首個工作移出，并繼續檢查，如果之後依然出現違反限制的工作，則繼續移出，直到

doi:10.6342/NTU201801942

所有工作都滿足限制式為止。

5 判斷部分

5.1 對於每個被擠出來的工作j如果$0 \leq g_j \leq g_{max}$，即這輪循環內工作沒有被排入過或者進入過循環 1 次，那麼將擠出部分步驟 4.2 中違反限制的工作移回工作池P。如果$g_j > g_{max}$個工作已經被排入過過至少兩次，則會與擠出部分開始前的結果進行比較，如果排序結果變差，則將結果初始化回沒有排入j_0的情況，并將j_0放入棄用池O，如果排序結果變好，則將違反限制的工作移回工作池P。

5.2 如果工作池P內依然有工作，則回到分配部分的步驟 3.1。如果工作池P內沒有工作，則將棄用池O的工作加入工作池，初始化工作池內和已經排到機器上的工作的循環數$g_j = 0$，并回到分配部分的步驟 3.1。如果回到第一步后發現沒有新工作，則排序結束。

我們可以看到，如果工作沒有最晚完成時間限制，如果我們將初始序列進行從長到短的排序，則該排序法會變為 Liu(2016)提出的 CHBF 排序法，如果有利益等於時間，則會簡化為 LPT 排序法。此外，在每台機器上，如果我們允許無限次循環，即$g_{max} = \infty$，則會在該機器上實現插入排序，如果存在一個 EDD 可行解，則這個排序會使機器上的工作以 EDD 排序。這樣我們就可以達成同時兼顧 LPT 以及 EDD 兩種排序法的排序方法。同時，我們要注意到由於工作往往不能在機器上找到 EDD 可行解，所以直接進行 EDD 排序反而會陷入該擠出哪個工作的困難選擇，這也是我們的算法沒有直接利用 EDD 可行解的原因。而在我們的算法中，我們可以保證擠出來的工作總體上是盡可能小的，這也是

doi:10.6342/NTU201801942

最差擠出算法的核心思想。

第三節 最差收益擠出算法具体例子

我們考慮一種情況，假如有 ABCDEF 六個工作被排入 2 臺機器里。各個工作的性質如下：

工作	A	B	C	D	E	F
利益	1	4	3	1	3	1
最晚完成時間	1	2	2	3	4	5
時長	1	2	2	2	3	3

表格 四.2 舉例試驗工作參數

首先如果直接使用 EDD 排序，則結果會是：

機器	依序被排入的工作	機器總利益
1	A	1
2	B F	5

表格 四.3 舉例試驗 EDD 排序結果

如果使用 CHBF（利益優先）排序

機器	依序被排入的工作	機器總利益
1	B	4
2	C	3

表格 四.4 舉例試驗 CHBF 排序結果

如果直接使用 LPT 排序

機器	依序被排入的工作	機器總利益
1	F	1

doi:10.6342/NTU201801942

2	E	3

表格 四.5 舉例試驗 LPT 排序結果

而如果採用 LBO 排序，在這裡由於工作數量較少，為了簡短步驟，取 $g_{max}=1$，$g_{max}=1$時，在一些極端情況下容易出現當一次性擠出多個工作時，陷入一個較差的解里，一般來說，取$g_{max}=2$時可以應對大多數情況。

總輪次	輪次 g	執行	結果
0		將工作 LDD 排序	J:ABCDEF
1		取出 2 個工作	J:CDEF P:AB
1		對工作池利益排序	P:BA
1	0	將 B 排入機器 1	1:B P:A
1		檢查	1:B 無擠出 P:A
1		對工作池利益排序	P:A
1	0	將 A 排入機器 2	2:A P:∅
1		初始化	B_1:4 B_2:1
2		取出 2 個工作	J: EF P: CD
2		對工作池利益排序	P:CD
2	0	將 C 排入機器 2	2:CA P:D
2		檢查	2:C A 擠出 P:DA
2		對工作池利益排序	P:A(0)D(0)
2	0	將 D 排入機器 2	2:DC P:A
2		檢查	2:D C 擠出 P:CA
2		對工作池利益排序	P:C(1)A(0)

doi:10.6342/NTU201801942

2	0	將 A 排入機器 2	2:AD P:C
2		檢查	無擠出
2		對工作池利益排序	P: C(1)
2	1	將 C 排入機器 2	2:CAD P:D
2		檢查	2:C AD 擠出 P:DA
2		對工作池利益排序	P:A(1)D(1)
2	1	將 A 排入機器 2	2:AC P:D
2		檢查	2:C A 擠出 O:A P:D
2		對工作池利益排序	P: D(1)
2	1	將 D 排入機器 2	2:DC P:∅
2		檢查	2:C D 擠出 O:AD P: ∅
2		初始化	B_1:4 B_2:3
3		取出 2 個工作	J: ∅ P: ADEF
3		對工作池利益排序	P:EADF
3	0	將 E 排入機器 2	2:EC
3		檢查	2:E C 擠出 P:ADCF
3		對工作池利益排序	P:C(1)A(0)D(0)F(0)
3	0	將 A 排入機器 2	2:AE P: CDF
3		檢查	2:AE 無擠出 P: CDF
3		對工作池利益排序	P:C(1) D(0)F(0)

doi:10.6342/NTU201801942

3	0	將 D 排入機器 1	1:DB P: CF
3		檢查	1:D B 擠出 P: CBF
3		對工作池利益排序	P:B(1)C(1) F(0)
3	0	將 F 排入機器 1	1:DF P: BC
3		檢查	1:DF P: BC
3		對工作池利益排序	P:B(1)C(1)
3	1	將 B 排入機器 1	1:BDF P: C
3		檢查	1:BF D 擠出 P: DC
3		對工作池利益排序	P:C(1)D(1)
3	1	將 C 排入機器 2	2:CAE P: D
3		檢查	2:AE C 擠出 P: D O:C
3		對工作池利益排序	P: D(1)
3	1	將 D 排入機器 2	2:ADE P: D
3		檢查	2:AE D 擠出 O:CD P: ∅
		排序完成	1：BF 2：AE B_1:5 B_2:4

表格 四.6 舉例試驗 LBO 排序過程以及結果

可見最差收益擠出算法成功的規避了 EDD 和 LPT 算法的弱點，找到了一組更為優秀的解。同時，我們可以注意到，結果為 AE 和 BF，其中，F 的效率實際上是最低的，而效率第二高的 C 也被排除在外。所以說，試圖直接找到一個「評分」來進行分配工作是相對困難的。但是相對來說，我們找到「評分」不及格的工作相對容易，找出當前違反限制式的工作即可。而通過擠出這一操作則可以大大減少系統的「試錯」

doi:10.6342/NTU201801942

時間。以一次排入擠出檢查為一個完整的步驟。這個流程中，我們總共進行了 14 次檢查。而我們總共有 6 個工作，不考慮順序共有 41 種分配情況。

第四節 最差收益擠出算法的時間複雜度

對於最差收益擠出算法的時間複雜度的計算，由於流程複雜，所以直接計算相對困難。但是我們可以用一個較為簡單的方法預估出其時間複雜度的上界。假設一個擁有著n個工作m臺機器的問題。在工作池分配階段，我們將進行n/m次步驟，哪怕改變分配方式，這個步驟也不會超過n次，因此這操作本身的耗時可以忽略不計，同理，我們可以忽略選擇工作分配到機器上這一過程所花的時間。在分配階段和擠出階段，我們可以知道工作池的大小不會超過n，而在某台機器上已經分配到的工作也同樣不會超過n。那麼假設最糟糕的情況，我們選出的一個工作分配到這台機器上時，這個機器上的工作正好是反向的 EDD 排序，而加入這個工作后則只能 EDD 才能排下。這個過程不會慢于時間複雜度為$O(n^2)$的插入排序法，而這個過程最多進行$n \cdot g_{max}$次。同時，每進行一次擠出，如果有新的擠出結果，則要進行一次工作池的排序，而這個步驟的本身的時間複雜度不會超過$O(n^2)$。因此，在某一工作池下，總步驟的時間複雜度不會超過$O(n^3)$。而由於工作池的更新次數不會超過n次，因此時間複雜度不會超過$O(n^4)$。在實作中，實際上不可能出現完成了整個排序后循環次數g只增加了一次的情況，而一般機器上以及工作池中的工作數量也不會那麼極端，排序時也不太會出現需要跑完整個過程恰好能夠加入的工作，所以實際耗時會遠遠低於上界。

doi:10.6342/NTU201801942

第五節 最差收益擠出算法的細節意義

在實現「最差收益擠出」這一過程中，我們採用了很多方法來保證「最差收益」能夠被順利的「擠出」。這些方法是啟發式的，很多是根據以往的經驗和實際試驗中遇到的極差情況而加入的步驟。儘管基本思想的部分和一些細節操作本身我們已經在第一節和第二節進行了詳解，我們將在這一節解釋這些細節操作的「為什麼」。事實上，這些細節的加入是最差收益擠出法得以達成的基礎之一，同時也大大提升了最差收益擠出法的穩定性。我們將對我們的擠出等步驟進行詳細的剖析，來展示我們為何需要加入這些步驟，已經這些步驟如何使得最差收益擠出法能達到一個頗為不錯的結果。

1　基本排序部分：

在擠出之前，我們需要一個方案來得到一個用來被「擠出」的排序結果。在最差收益擠出法中，這一結果通過兩部分來達到。其一是決定工作被排序先後的順序，其二是決定工作分配的順序。本文選用 LDD 的原因是其較難被擠出，而使用 HBF 則是根據 Liu（2016）對於相關問題的經驗。這部分的研究還有很多的試驗和研究可以進行，根據具體的問題可能也會有相對更合適的方案，我們選取的只是一種啟發式的中庸的選擇，我們將會在數值試驗部分看到一些定性的結果。由於本文的主要目的是引入「擠出」這一概念並將其實用化，因此不再對這方面內容進行更多的探討。

2　工作排入部分的選位與擠出問題

工作被排入后，我們選擇將其加入一個「合適」的位置并給出了兩條規則。其一是不能違反其本身的相關限制式，這個的目的是保證工作被排入。這一思想來源于退火算法，我們希望能夠允許目標值的暫時降低來獲取更多的可能。由於最後截止時間限制的關係，在某一台機器上

doi:10.6342/NTU201801942

很容易出現無法加入新的工作的情況，這一結果並不一定是足夠優秀的，因此我們強制新的工作在違反限制式之前加入隊列，來讓這個結果進行改變，獲得更多的可能。其二是機器上前一個工作的目標值不能比當前工作小。這一目的是在工作發生被擠出時，盡可能的擠出比較小的工作。我們知道從機器上選擇「最合適」的擠出工作可以被視為是把工作分為兩類，讓其中一邊結果最大，是一個 partition 問題的變種。這是一個 NP-hard 問題，這也就意味著我們必須找出一個相關的排序法來解決這個問題。但是，傳統的 partition 問題的排序法針對的是等分，而我們這裡並不相同。因此我們採用的思路是讓所有的工作的排序本身是收益越低靠後。由於對於同一個最晚完成時間，越靠後就越容易違反限制式，我們讓收益較低的工作靠後，就更容易實現「最差收益擠出」。誠然，這裡有個矛盾，大的工作排在前面雖然更難以違反限制式，但是如果多個工作違反限制式時，由於我們的擠出規則是從第一個工作開始逐步向後判斷，相對較大的工作可能會由於排序較為靠前先被擠出，然後小的工作就不會違反限制式了。對於這一問題，我們可以假設當一個工作被擠出時，我們不直接擠出而是進行交換，易知只有和其之前的工作進行交換才可能改變其被擠出的結果。而在這個工作之前的工作只有比這個工作收益更大或者最晚工作截止時間更早的工作。對於前者，按照最差收益擠出的思路，肯定是當前被擠出的工作更適合作為擠出對象。而對於後者，這也是我們算法加入循環的原因。在下一次循環中，如果此時依然是這個被擠出的工作加入了這台機器，那麼根據不能比後面的工作收益小的原則，這個工作將會被加入到那個之前沒有被擠出最晚工作截止時間更早的工作之前，從而讓他可以擠出這些較小的工作。這也引出了最差收益擠出法的一個可能的最差情況，即一個收益足夠大但是工作所需的完成時間也過長的工作，可能會像石頭一般「沉」工作隊列的前

doi:10.6342/NTU201801942

方，而之後的排序過程都集中于對于該工作之後工作的改善。這一問題可能是最差工作擠出法函待解決的一個重要的最差情況。剔除工作本身，適當的調整工作排入的順序使得在這個工作之後能夠有擠出該工作的工作都是可能的解決辦法。

3　算法的循環問題

從排入和擠出問題的討論中，我們可以得知，為了避免算法陷入局部最優，我們強制讓新的工作排入。而這一步驟自然可能導致目標值的下降。因此，設置適當的循環方案也是保證算法結果的手段。如果一個工作的排入并擠出另外一個工作后導致了目標值的下降，那麼接下來的排序中，根據 HBF 規則，我們依然會改善同一台機器。如果在排入一個或者數個新的工作后目標值上升，我們則完成了尋找其他最優的過程。如果沒有改善，那麼可以預計的是我們之前被擠出的工作又會被排入這台機器中。根據排入法則，這個曾經被擠出過的工作會排到那個擠出他的工作之前。這樣就實現了工作的前後換位，從而出現可能的兩個工作都排入的結果。但是這一過程可能會導致無限循環。因此我們選擇加入工作循環最大次數，通過當工作超過循環最大次數限制時只取使目標值變好的排序結果，我們就可以讓這一進程總體上使得目標值上升。同時在排序中每加入新工作便初始化工作本身的循環次數，這樣讓每個工作有更多的「挑戰」機會。

4　額外循環問題

在實作中，排序接受后我們通過加入 0 收益工作來增加額外的循環次數，這原因在於我們之前用盡循環次數，被認為是該輪「不重要」的工作，可能並不是排序結束時目標值最小的機器的「不重要」的工作，這些工作可能在之前改善其他機器的過程中用盡了循環次數。因此我們加入額外的循環，就可以讓這些工作獲得在當前機器上再試一次的機會。

doi:10.6342/NTU201801942

事實上，在第五章的數值研究中，我們對一些特定情況的工作加入數量進行了研究，那裡的結果也是在基本的循環完成后，額外加入一組 0 收益工作進行了再一次循環的結果。

doi:10.6342/NTU201801942

第五章 數值試驗

第一節 試驗介紹與試驗參數

使用數值試驗是驗證一種算法有效性的常用方法。在這一章中，我們對 6 種情況進行了 114 種不同類別的數值試驗。每一次試驗將平均 100 次的結果來與整數規劃所得的最佳解、線性規劃所得的解上界、基因算法所得的結果進行比較。具體參數如下：

試驗參數表

試驗參數	數值
n	{15, 30, 50, 100, 150}
m	{3, 10, 30}
c_j	[0, 100]
K_i	{200, 300, 600, 1800, ∞(10000)}
b_j	$\{c_j,\ {c_j}^2,\ \sqrt{c_j}$, rand, rand$(\sqrt{c_j}, {c_j}^{5/4})\}$
D_j	$\{3c_j,\ 4c_j,\ 8c_j,\ 27c_j,\ \infty(1000c_j)$, rand, rand$(c_j\frac{2n}{m}\cdot c_j)\}$

表格 五.1 數值試驗參數表

對於試驗樣本的參數，我們將考察不同n/m比的工作的同時，考慮不同機器工作時長限制、不同利益曲線和不同最晚截止時間的情況。主要考察的參數的數值來源為：

D_j最晚完成時間限制。較緊(DT)的情況下統一設計為 3 倍，較寬鬆(DL)的情況下則根據n/m比設計為 4~27 倍不等。此外還有無D_j限制(DK)以及額外兩種隨機的情況。第一種是隨機情況是不超過 DL 的較為緊張的情況 DRT，另外一種是開源最大不超過$2n/m$倍的較為寬鬆的情況(DRL)。

K_i機器的最大工時限制。除了無限制(KK)外，統一設計為$100\cdot(D_j -$

doi:10.6342/NTU201801942

1)，即根據當前試驗的D_j參數倍率來決定K_i，在實驗中這種情況標記為KD。

b_j為利益的倍率。試驗中我們將會用到線性 bL，凹 bA，凸 bX 以及兩種隨機情況。第一種是在$[0, 100]$的範圍內即c_j的取值範圍內隨機產生(R)，另外一種是考慮到我們之前的試驗中重點研究了凹凸和線性關係對算法表現的影響，因此隨機的範圍將根據c_j在$[\sqrt{c_j}, {c_j}^{5/4}]$內取值，這個範圍的平均值依然接近$c_j$，這種隨機我們標記為 RC。

第二節 數值試驗

數值試驗中我們將把基因算法作為我們的參照對象。基因算法在諸多應用中已經證明了其價值。在我們這個問題上是一個良好的參照對象。我們的試驗中，基因算法的參數為：種群數目 50，交叉概率為 0.6，變異概率為 0.1，統一進化 2000 代并選擇其中最好的結果。除此以外，我們也將通過整數規劃來計算其最優解，對於工作數量較多的情況，由於這是個 NP-hard 問題，我們將依靠線性規劃來求出解的上界來作為參照。得益于我們在模型構建時的關於D_j限制式在默認 EDD 初始排列下有效的證明，線性規劃下的模型只需要去除x_{ij}的整數設定即可，其意義在於分配給機器的工作可以被分割。我們通過使用 AMPL 程式中的 CPLEX 作為我們求解整數規劃以及線性規劃問題的手段。在試驗結果中，如果沒有列出整數規劃的結果，則說明 AMPL 無法在短期內求解這個整數規劃問題。

關於基因算法，我們採用的基因是一個長度為n取值為$[0,m]$的數組，並且在初始化時通過「剪除」不符合限制條件的工作以保證可以產生一組合適的初始基因，否則該算法將會花費過多的時間在初始化階段。由於基因算法在 2000 代時花費的時間已經不短於最差收益擠出法，在工

doi:10.6342/NTU201801942

作量較大時更是遠遠超過最差收益擠出法，故統一採用 2000 代作為試驗的標準。

工作與機器數量試驗結果表

n	m	LBO/IP	GENE/IP	LBO/LP	GENE/LP	LBO/GENE
15	3	0.931798	0.945271	0.906976	0.918363	0.988122
30	3	NP	NP	0.917569	0.898762	1.032788
100	3	NP	NP	0.918288	0.840305	1.13213
50	10	NP	NP	0.896556	0.80085	1.1375
100	10	NP	NP	0.905344	0.788975	1.194936
150	30	NP	NP	0.889958	0.645268	1.418141

表格 五.2 工作與機器數量試驗結果表

最差利益擠出法整體來說表現較為穩定，整體上來說要遠遠優於基因算法。對於 15 個工作的情況，基因演算法跑完了 2000 代也僅僅稍好于最差利益擠出法，而通過查詢試驗記錄可知，於此同時最差利益擠出法僅僅進行了 32 個步驟。

在工作稍多時，我們已經無法計算出整數規劃的最佳解，通過第一組數據的比較，我們有理由相信最差利益擠出法總體表現可以達到最佳解的 90%，而在 30 臺開始，最差收益擠出法的效果只是在緩慢下降，已經相對於基因算法有了明顯的優勢。

機器產能限制試驗結果表

K_i	LBO/IP	GENE/IP	LBO/LP	GENE/LP	LBO/GENE
KK	0.942288	0.956862	0.917367	0.830507	1.128881
KD	0.917373	0.929333	0.889853	0.794677	1.18047

表格 五.3 數值試驗機器產能試驗結果表

我們可以看到整體表現上，最差利益擠出法是要優於基因演算法的。

doi:10.6342/NTU201801942

雖然在 IP 的結果上稍差，但是根據我們對於機器和工作數量的研究，IP 情況下實際上只有工作數量為 15 個機器為 3 臺可以整數規劃求解。基因算法僅僅在工作量非常少時表現才能稍勝一籌。對於更多更複雜的情況，最差收益擠出法是要明顯優於基因算法的。

工作截止時間試驗結果表

D_j	LBO/IP	GENE/IP	LBO/LP	GENE/LP	LBO/GENE
DT	0.908689	0.917743	0.848642	0.732844	1.233546
DL	0.930358	0.955459	0.915809	0.847543	1.102285
DK	0.982813	0.966366	0.986636	0.873497	1.145776
DRT	0.873488	0.940541	0.82309	0.749193	1.153117
DRL	0.907153	0.954196	0.909136	0.857633	1.073181

表格 五.4 數值試驗工作截止時間試驗結果表

可以看到，在無截止時間限制時，其結果已經非常接近最優解。這說明我們在沒有截止時間限制式時，希望能夠達到 CHBF 表現的目標已經達成。而在考慮到截止時間限制的其他情況，整體表現也遠超遺傳算法，可以做到整數規劃結果的 90%。其中 DT 和 DRT 的情況較差，這可能是因為限制式越緊，可選解在所有排列中的比例就越少，那麼擠出上限就會更多的被達到，從而影響了結果。

工作利益試驗結果表

b_j	LBO/IP	GENE/IP	LBO/LP	GENE/LP	LBO/GENE
bL	0.952396	0.95398	0.944604	0.855807	1.114589
bA	0.956478	0.927103	0.931921	0.747838	1.339563
bX	0.9197	0.953052	0.872556	0.852222	1.027619
R	0.907153	0.954196	0.909136	0.857633	1.073181
RC	0.873488	0.940541	0.82309	0.749193	1.153117

doi:10.6342/NTU201801942

表格 五.5 數值試驗工作利益試驗結果表

對於不同工作利益的情況，試驗結果表明凹函數的情況表現較好而凸函數的情況表現較差，但總體而言依然保持著一個較好的結果。證明擠出收益最差的工作這一做法是解決這類多限制式問題的一個比較好的選擇。

本節試驗的具體數據可以在附錄中查閱。

第三節 參數估計

作為一個啟發式算法，最差收益擠出法的參數也會影響到其表現。正確的使用參數也會影響到算法的結果，由於試驗條件限制，我們無法對所有的參數進行實證研究其影響，因此只能選擇一些特殊情況來展示參數調整的價值已經闡述我們選擇這個參數的理由。

最大代數g_{max}影響到算法的時間複雜度，但是如果g_{max}太低，則會導致工作在擠出過程中過早的被放棄。因此調整g_{max}會影響擠出算法的效果。我們這裡選擇比較複雜的情況之一：150 個工作 30 臺機器 RC&DRT&KD 的情況取 30 次試驗的平均。這種情況下 LBO/LP 數值較低，原因除去實際 IP 的結果可能遠小於 LP 的結果以外，合理的解釋是在這種情況下工作會比較容易被擠出，而每個機器可供改善的機會較少。如果我們增減g_{max}，其改善效應將會比較明顯。

g_{max}	1	2	3	5	10
LBO/LP	0.817042	0.837998	0.839067	0.837296	0.833157

表格 五.6 循環代數參數估計試驗結果表

可見，g_{max}增加時，改善的效果並不明顯。選擇$g_{max} = 2$遍已經可以達到足夠好的結果。此外我們發現在$g_{max} = 1$時，出現極端差情況的幾率要大大高於$g_{max} = 2$。值得注意的是，這個試驗針對的是特定的隨機出來的 30 種情況，因此肯定是有所偏頗的。如果在實作中發現最差

doi:10.6342/NTU201801942

收益擠出算法出現極端差情況時，可以增大g_{max}來獲得一個不同的結果。

工作池加入的工作數量越多，理論上我們每次選擇工作進入機器時，可以參照的工作也就越多。而工作池加入的工作越少，循環執行的次數就越多，這都可能對結果產生改善。

加入工作數	1	m	$2m$	$5m$(max)
LBO/LP	0.85024	0.837998	0.858629	0.887962

表格 五.7 加入工作池參數估計試驗結果表

我們可以看到，如果一次性把所有工作都加入進去效果是最好的。這個結論還需要更多的試驗加以驗證，但是毫無疑問改變工作池加入的工作數量是一種算法可行的改善方案。

不同的初始排序法也會對最差擠出排序法的結果產生影響。目前本算法是使用 LDD 來進行初始化，除此之外，我們考察了一些其他的情況，例如 LDD（最晚工作截止時間優先），HBF（最高利益優先），LBF（最低利益優先）等。其中 HBLSF 意義為最高收益最低空間排序。即以$b_j/(D_j - c_j + \lambda)$為標準的排序。通俗的講就是利益高但是又容易被擠出的工作先排進去，LBHSF 則相反。

初始排序方案	EDD	LDD	LBF	HBF	LBLSF	HBLSF
LBO/LP	0.809502	0.837998	0.777075	0.88025	0.782434	0.847814

表格 五.8 不同初始排程方法試驗結果表

我們可以看到，不同的初始排序順序也會對算法產生影響。EDD 和 LBF 偏差而 HBF 偏好。由此可見，選擇初始排序法時，可能參照盡量讓目標式優先或者盡量讓主要限制式影響的工作靠後可能是比較合理的選擇。由於這個試驗暫時只針對一種情形，具體的選擇可能還需要根

doi:10.6342/NTU201801942

據其需要解決的問題的數據特性來決定。目前採用的 LDD 作為初始排序是一個折衷的選擇。

作為對照條件的基因算法也會受到一些參數的影響，我們在這裡不會去調整其變異幾率等參數，只考察其遺傳代數對結果的影響。本研究中，我們設定了基因演算法的最大迭代次數，這樣就使得我們統計耗時意義較小。這裡我們通過統計不同迭代次數下基因演算法的表現，可以對工作較多時，基因算法要運行多久就可以達到最差收益擠出法的表現提供一個定性的參考。

代數	500	2000	5000	50000	200000*
LBO/GENE	2.278313	1.849428	1.615167	1.29223	1.227784

表格 五.9 基因算法以及 LBO 耗時估計

可見，在 150 個工作 30 臺機器的情況下，基因演算法在 50000 次迭代后依然遠不如最差收益擠出法的結果。在 20 萬次時，我們進行了 5 次試驗，僅有 1 次達到接近最差收益擠出法的結果。總體來看，在工作數量較多的情況下，如果要讓基因算法達到和最差收益擠出法相當甚至更好的結果，還需要大量參數調配以及數倍的運行時間。

doi:10.6342/NTU201801942

第六章 總結和未來計劃

最差工作擠出法是一個全新的針對考慮公平性、不同機臺產能與不同工作截止時間的工作分配問題的排序法。對於這個排序法本身，我們還有很多需要研究的地方，例如參數和基礎排序法的選擇現在還依賴於研究者在設計算法時的經驗，在實作中也許還可以有更好的選擇。此外，數值試驗畢竟只能反映一些預設的情況，對於真實情況可能會更為複雜多變，這個排序法的實用性也需要在真實的環境中進行更多的檢驗。但是僅僅就試驗結果而言，相對於同時考慮公平性、機台總產能、工作截止時間的複雜問題而言，這個演算法的表現令人驚喜。在工作數量較多時，這個算法依然能夠保持極高的效率和較好的結果，非常值得嘗試在實務中進行實用。

最差工作擠出法自 LPT 算法開始一脈相承，再次證明了對於公平分配問題，把目標結果影響較小的工作放在「最後」是一種非常實用的思想。針對這一思想，最差工作擠出法還有很多的潛力可以挖掘。例如我們可以嘗試加入工作的最早開始時間，或者在機器中加入「阻礙」，譬如某台機器在某一時刻開始將會有一段時間不能運行。同時，就目前的算法而言，機器效率也可以被積極的考慮進去，引入機器效率並不會影響排序的流程，甚至可以讓機器的利益折算和機器的效率取不同的值。這些限制條件雖然每個都會極大的改變問題本身，但是並不會影響最差工作擠出法的「擠出」這一規則。有理由相信最差收益擠出法可能可以在這類問題上有用武之地。

跳出時間限制，最差收益擠出法的「最差收益擠出」可能在其他情形上也可以適用。例如如果要同時考慮運費、空間以及車輛載重的運輸問題等。對於所有存在不相干限制式的問題，最差目標值擠出都可能是

doi:10.6342/NTU201801942

一種可行的啟發式解決方案。我們也將會就這個算法在更多的問題上進行嘗試。

最後就算法本身而言，在基本設計中，算法的弱點是如果遇到一個比較大的工作，這個工作可能會沉在最差收益的機台的工作隊列前方，導致無法有效的擠出工作。這一點一方面可以通過文中所述的調整參數來進行改善，例如比這個工作的截止時間更早或者收益更大的工作都可以擠出這個工作，那麼調整初始排序就可以改進這個問題。在一些研究中，已經開始採用人工智慧來協助工作分配。就最差收益擠出算法而言，尋找「搗亂」的那個工作也許也可以交給人工智慧來協助完成。比起直接在全部的排序可能中尋找線索，利用最差收益擠出來完成一些基礎的排序可能是一個更好的選擇。

doi:10.6342/NTU201801942

參考文獻

Csirik, J., H. Kellerer, G. Woeginger. 1992. The exact LPT-bound for maximizing the minimum completion time. *Operations Research Letters* **11**(5) 281-287.

Deuermeyer, B.L., D.K. Friesen, M.A. Langston. 1982. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM Journal on Algebraic Discrete Methods* **3**(2) 190-196.

Graham, R.L. 1966. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal* **45**(9) 1563-1581.

Graham, R.L. 1969. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics* **17**(2) 416-429.

Liu, C.-W., 2016. *Job Allocation with a Consideration of Fairness*, master thesis, National Taiwan University, Taiwan.

Pinedo, M. 2012. *Scheduling Theory: Algorithms and Systems.* Springer, Berlin, Germany.

Tsai, L. H. 1992. An algorithm for flow time minimization and its asymptotic makespan properties. *Information processing letters*, **43**(1), 41-46.

Tsai, L. H. 1992. Asymptotic analysis of an algorithm for balanced parallel processor scheduling. *SIAM Journal on Computing*, **21**(1), 59-64.

He, Y., Tan, Z., Zhu, J., Yao, E. 2003. κ-Partitioning problems for maximizing the minimum load. *Computers & Mathematics with Applications*, **46**(10-11), 1671-1681.

Walter, R. 2013. Comparing the minimum completion times of two longest-

doi:10.6342/NTU201801942

first scheduling-heuristics. *Central European journal of operations research*, **21**(1), 125-139.

Lawler, E. L., & Moore, J. M. 1969. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, **16**(1), 77-84.

Bertsimas, D., V.F. Farias, N. Trichakis. 2011. The price of fairness. *Operation Research* **59**(1) 17–31.

doi:10.6342/NTU201801942

附錄

參數	
n	工作的數量
m	機器的數量
K_i	機器的工作時間限制 $(K_i > 0)$
c_j	工作的時間長度 $(c_j > 0)$
b_j	工作的利益 $(b > 0)$
D_j	工作的最晚完成時間 $(D_j > 0)$

以上參數為下表中涉及的參數，其中c_j為取值[0,100]的隨機數。K_i為固定值，b_j與D_j為K_i的倍數，其餘標識詳見 27 頁第五章第一節。

試驗具體結果見下表：

n	m	D_j	b_j	K_i	LBO	GENE	IP	LP
15	3	3	1	10000	218.31	224.30	235.94	246.72
15	3	3	1	200	186.00	183.13	198.33	199.53
15	3	3	0.5	10000	27.38	30.07	32.01	33.12
15	3	3	0.5	200	23.57	25.12	26.69	28.08
15	3	3	2	10000	14376.06	14383.18	15631.91	16183.38
15	3	3	2	200	12510.80	11178.23	13436.79	14556.43
15	3	4	1	10000	239.71	241.62	249.79	249.89
15	3	4	1	300	244.41	245.56	255.73	256.14
15	3	4	0.5	10000	31.43	32.12	33.41	33.43
15	3	4	0.5	300	31.11	31.67	32.97	33.28
15	3	4	2	10000	16448.65	16328.71	16896.95	16927.50
15	3	4	2	300	15848.24	15524.05	16351.74	16461.15

doi:10.6342/NTU201801942

15	3	1000	1	10000	243.07	239.33	247.14	247.23
15	3	1000	0.5	10000	32.63	32.20	33.43	33.45
15	3	1000	2	10000	16467.96	16114.40	16651.65	16662.26
30	3	3	1	10000	260.92	247.30	N/A	290.47
30	3	3	1	200	192.47	184.31	N/A	200.00
30	3	3	0.5	10000	31.81	35.14	N/A	42.85
30	3	3	0.5	200	23.88	28.49	N/A	33.24
30	3	3	2	10000	19775.45	16042.48	N/A	23677.91
30	3	3	2	200	14319.91	9648.34	N/A	17546.79
30	3	8	1	10000	493.56	495.47	N/A	513.39
30	3	8	1	600	492.05	494.46	N/A	513.73
30	3	8	0.5	10000	62.45	63.59	N/A	66.03
30	3	8	0.5	600	63.03	64.02	N/A	66.68
30	3	8	2	10000	32400.76	31864.18	N/A	32959.83
30	3	8	2	600	32947.04	32458.23	N/A	33746.81
30	3	1000	1	10000	492.95	475.31	N/A	494.98
30	3	1000	0.5	10000	65.55	63.40	N/A	66.02
30	3	1000	2	10000	34187.46	33106.10	N/A	34249.72
100	3	3	1	10000	280.26	252.43	N/A	297.04
100	3	3	1	200	193.61	174.03	N/A	200.00
100	3	3	0.5	10000	37.02	40.47	N/A	50.52
100	3	3	0.5	200	25.85	32.40	N/A	40.02
100	3	3	2	10000	25565.94	14239.13	N/A	28241.72
100	3	3	2	200	14729.67	7078.15	N/A	19286.41

doi:10.6342/NTU201801942

100	3	27	1	10000	1565.01	1506.90	N/A	1672.90
100	3	27	1	1800	1550.77	1502.56	N/A	1655.40
100	3	27	0.5	10000	203.47	199.16	N/A	221.39
100	3	27	0.5	1800	203.98	198.34	N/A	221.90
100	3	27	2	10000	106661.1	101258.9	N/A	110223.3
100	3	27	2	1800	108140.1	102970.5	N/A	111500.6
100	3	1000	1	10000	1657.10	1498.57	N/A	1657.70
100	3	1000	0.5	10000	221.68	198.08	N/A	222.00
100	3	1000	2	10000	113154.3	103996.7	N/A	113159.5
50	10	3	1	10000	215.41	202.69	N/A	249.53
50	10	3	1	200	187.36	158.57	N/A	199.99
50	10	3	0.5	10000	27.87	27.98	N/A	33.35
50	10	3	0.5	200	23.44	22.81	N/A	28.15
50	10	3	2	10000	14772.25	11891.01	N/A	16777.94
50	10	3	2	200	13371.89	7962.41	N/A	15213.95
50	10	4	1	10000	235.99	214.64	N/A	248.41
50	10	4	1	300	236.59	218.83	N/A	248.98
50	10	4	0.5	10000	30.68	28.82	N/A	33.29
50	10	4	0.5	300	30.57	29.46	N/A	33.42
50	10	4	2	10000	16795.58	14607.37	N/A	17110.24
50	10	4	2	300	16401.43	13798.52	N/A	16820.62
50	10	1000	1	10000	246.29	216.67	N/A	250.75
50	10	1000	0.5	10000	32.03	28.50	N/A	33.43
50	10	1000	2	10000	16638.86	14494.72	N/A	16754.27

doi:10.6342/NTU201801942

100	10	3	1	10000	259.50	212.07	N/A	296.48
100	10	3	1	200	190.68	152.56	N/A	200.00
100	10	3	0.5	10000	31.12	32.25	N/A	44.12
100	10	3	0.5	200	22.71	25.71	N/A	33.35
100	10	3	2	10000	20269.87	11929.31	N/A	24575.74
100	10	3	2	200	15763.36	6641.22	N/A	17763.76
100	10	8	1	10000	475.98	437.39	N/A	501.48
100	10	8	1	600	473.73	434.66	N/A	498.11
100	10	8	0.5	10000	61.78	58.08	N/A	66.76
100	10	8	0.5	600	61.69	58.29	N/A	66.68
100	10	8	2	10000	32769.16	29357.52	N/A	33734.72
100	10	8	2	600	32484.22	29032.36	N/A	33371.65
100	10	1000	1	10000	498.46	437.68	N/A	500.82
100	10	1000	0.5	10000	65.56	58.05	N/A	66.56
100	10	1000	2	10000	33139.98	28764.62	N/A	33168.74
150	30	3	1	10000	213.50	178.36	N/A	248.14
150	30	3	1	200	188.32	139.01	N/A	200.00
150	30	3	0.5	10000	27.79	24.98	N/A	33.39
150	30	3	0.5	200	23.46	20.81	N/A	28.23
150	30	3	2	10000	14604.38	9850.95	N/A	16652.59
150	30	3	2	200	13493.52	6075.43	N/A	15134.70
150	30	4	1	10000	234.52	177.10	N/A	250.19
150	30	4	1	300	235.66	185.72	N/A	250.12
150	30	4	0.5	10000	30.14	23.93	N/A	33.27

doi:10.6342/NTU201801942

150	30	4	0.5	300	30.10	25.00	N/A	33.28
150	30	4	2	10000	16127.88	10667.36	N/A	16611.00
150	30	4	2	300	16004.24	10628.95	N/A	16470.83
150	30	1000	1	10000	244.25	171.81	N/A	250.03
150	30	1000	0.5	10000	31.39	23.67	N/A	33.32
150	30	1000	2	10000	16483.08	10436.76	N/A	16561.00
15	3	DRT	RC	10000	173.39	182.99	223.35*	218.53
15	3	DRT	RC	300	160.93	171.48	225.27*	214.42
30	3	DRT	RC	10000	395.89	396.45	N/A	432.02
30	3	DRT	RC	600	386.30	393.44	N/A	431.75
100	3	DRT	RC	10000	1395.39	1314.77	N/A	1454.39
100	3	DRT	RC	1800	1392.72	1318.52	N/A	1454.71
50	10	DRT	RC	10000	154.25	146.01	N/A	211.33
50	10	DRT	RC	300	159.43	135.86	N/A	222.52
100	10	DRT	RC	10000	379.63	348.05	N/A	432.01
100	10	DRT	RC	600	376.89	339.04	N/A	433.30
150	30	DRT	RC	10000	151.31	92.30	N/A	212.83
150	30	DRT	RC	300	155.45	87.39	N/A	219.55
15	3	DRL	R	10000	231.38	241.34	253.04	255.17
15	3	DRL	R	300	220.11	233.49	244.59	247.35
30	3	DRL	R	10000	472.58	476.27	N/A	499.47
30	3	DRL	R	600	477.79	479.17	N/A	504.37
100	3	DRL	R	10000	1617.78	1507.83	N/A	1679.14
100	3	DRL	R	1800	1617.34	1507.53	N/A	1681.83

doi:10.6342/NTU201801942

150	30	DRL	R	10000	217.62	166.81	N/A	251.67
150	30	DRL	R	300	210.30	158.68	N/A	248.76
100	10	DRL	R	10000	461.88	429.31	N/A	497.37
100	10	DRL	R	600	458.51	426.40	N/A	497.07
50	10	DRL	R	10000	219.72	211.27	N/A	248.96
50	10	DRL	R	300	213.20	208.92	N/A	250.60

*該數據只有部分 IP 結果，故平均值會比 LP 的結果要大。

試驗所用的源代碼可以在 GitHub 上下載，項目名 leastbenefitout。

doi:10.6342/NTU201801942