The Price of Approximate Stability for Scheduling Selfish Tasks on Two Links

Eric Angel [1], Evripidis Bampis [1] and Fanny Pascual [1]

Abstract. We consider a *scheduling game*, where a set of n selfish agents (traffic loads) want to be routed in exactly one of the two parallel links of a system. Every agent aims to minimize *her own completion time*, while the social objective is the *makespan*, i.e. the time at which the last agent finishes her execution. We study the problem of optimizing the makespan under the constraint that the obtained schedule is a (pure) Nash equilibrium, i.e. a schedule in which no agent has incentive to unilaterally change her strategy (link). This model, known as the CKN model, was first studied in [1] in the framework of *coordination mechanisms.* We show that, contrary to the KP model [2], there is not always an optimal solution which is a Nash equilibrium in the CKN model. For this reason, we consider a relaxation of the notion of *equilibrium* by considering α-approximate Nash equilibria where an agent does not have *sufficient* incentive (w.r.t. the value of α) to unilaterally change her strategy. Our main contribution is the study of the *tradeoff* between the *approximation ratio* for the makespan and the value of α. We first give an algorithm which provides a solution with an approximation ratio of $\frac{8}{7}$ for the makespan and which is a 3-approximate Nash equilibrium, provided that the local policy of each link is *Longest Processing Time* (LPT). Furthermore, we show that a slight modification of the classical *Polynomial Time Approximation Scheme* (PTAS) of Graham allows to obtain a schedule whose makespan is arbitrarily close to the optimum while keeping a constant value for α. Finally, we give bounds establishing relations between the value of α and the best possible value of the approximation ratio, provided that the local policies of the links are LPT.

1 Introduction

The scheduling setting that we consider in this paper is the following one: we are given a simple network of two parallel links and a set of n selfish agents. Each agent has some (positive) traffic load and wants to use exactly one of the parallel links to route it through. Equivalently, every agent can be viewed as a task, and each link as a machine (or processor). Every agent aims to maximize her own profit, and there are two basic models depending on what is considered as the individual profit of the agents:

- In [2], the *profit* of an agent is the inverse of the completion time of the machine on which she is assigned to. This model is known as the KP model.
- In [1], the *profit* of an agent is the inverse of her completion time. This model is known as the CKN model.

[1] IBISC, CNRS FRE 2873 – Université d'Évry Val d'Essonne, –, 523 Place des Terrasses, 91000 Évry, France. E-mail: {angel, bampis, fpascual}@lami.univ-evry.fr

In both cases, the *social optimum (or global objective function)* is the *makespan*, i.e. the time at which the last agent terminates her execution.

Our aim is to obtain a schedule which minimizes the makespan and which at the same time is *stable*, i.e. such that no agent has incentive to unilaterally change her link. More precisely, we assume that each processor uses a local policy (known by all the agents) in order to schedule the tasks assigned to it, and if a task decides to leave, in the proposed schedule, its current machine to go on another machine, knowing its local policy, it can calculates its new completion time. Therefore, the solution we seek is a *best Nash equilibrium* w.r.t. the global objective function. A new measure for evaluating the impact of searching a stable solution, i.e. a Nash equilibrium, has been introduced by Schultz et al. [3] and by Anshelevich et al. [4]: the *price of stability* is defined as the ratio of the objective function in a *best* Nash equilibrium and a global optimum schedule (this maximum is taken over all solutions). This measure can be viewed as the *optimistic* price of anarchy [2].

In the KP model there always exists a pure Nash equilibrium which is an optimal solution w.r.t. the makespan, and so the price of stability is 1 for this model. This result is a direct corollary of the following fact: it is always possible in the KP model to transform (*nashify*) an initial solution into a pure Nash equilibrium without increasing the value of the makespan [7]. It is then natural to ask if this is also possible for the CKN model. Notice that the the price of stability depends on the local policies of the links. We assume that each link schedules the tasks assigned to it using the Longest Processing Time (LPT) order, i.e. each link schedules its tasks from the largest one to the smallest one. In that case, it is not difficult to see that there is only one pure Nash equilibrium and that this schedule can be obtained using the classical LPT list scheduling algorithm. This shows that the price of stability in that case is 7/6, i.e. the approximation ratio of the LPT algorithm [6].

A natural way to improve this ratio, is to relax the definition of a "stable" schedule. We consider that a schedule is stable if it is an *α-approximate Nash equilibrium* ($\alpha \geq 1$), i.e. a schedule in which no agent has *sufficient incentive* to unilaterally change its behavior. We say that an agent does not have *sufficient incentive* to unilaterally leave the link on which it is scheduled, if and only if this does not increase its profit by more than α times its current profit[2]. We can define now the *price of α-approximate stability* as the ratio of the objective function in a *best* α-approximate Nash equilibrium and a global optimum schedule (this maximum is taken over all solutions). If $\alpha = 1$ we get the price of stability defined before. Thus, if the price of α-approximate stability is γ, it means that no algorithm (even an exponential one) with an approximation ratio smaller than γ can return schedules which are always α-approximate Nash equilibrium.

[2] Note that this definition is different from the definition of an ϵ-Nash equilibrium in [5]: in [5], if a solution is an ϵ-Nash equilibrium, then if an agent unilaterally changes strategy, her profit should be smaller than or equal to her current profit *plus* ϵ (and not *times* α as in our definition).

Example: Let us consider two machines using LPT policy, and the following tasks: two tasks t_1 and t_2 of length 3 and 3 tasks t_3, t_4 and t_5 of length 2. The only pure Nash equilibrium is the schedule where the tasks of length 3 are scheduled at time 0, and are followed by the tasks of length 2. This solution has a makespan of 7, whereas the makespan of the optimal solution is 6. In the optimal solution, let us consider the task of length 3 which starts at time 3. By going on the second machine, this task would be the largest one of this machine and would then be scheduled in the first position. Its completion time would then be 3 instead of 6. Since this task can reduce its completion time by a factor of 2 by changing machine, we will say that this task is *2-approximate*. Each task of this schedule can reduce its completion time by at most 2 by changing machine, so this schedule is a 2-approximate Nash equilibrium.

Our contribution: We give relations between α and the approximation ratio for the makespan, provided that the policies of the links are LPT. We show that the price of α-approximate stability is at least $\frac{8}{7}$ for all $\alpha < 2.1$, and at most $\frac{8}{7}$ for all $\alpha \geq 3$. We give in Section 2 an algorithm which shows this last bound: it achieves an approximation ratio of $\frac{8}{7}$ for the makespan and returns a 3-approximate Nash equilibrium. We show that the price of α-approximate stability is larger than or equal to $1+\varepsilon$ for any $\alpha \leq k$, where k is a certain constant in $\Theta(\varepsilon^{-1/2})$, and that it is smaller than or equal to $1+\varepsilon$ for any $\alpha \geq \frac{1}{\varepsilon}$ (see Section 3). This last bound is obtained by analyzing the PTAS of Graham (slightly modified) [6]. Section 4 shows a summary of the negative and positive results of this paper.

2 A variant of LPT

For simplicity in the sequel, we adopt the classical terminology of scheduling theory. We want to schedule n tasks $\{t_1, \ldots, t_n\}$ on 2 identical machines, and we want to minimize the maximum completion time of the last task scheduled, i.e. the *makespan*. We denote by P_1 the machine with maximum load, and by P_2 the other machine. Let n_i be the number of tasks scheduled on P_i. Let x_i be the i^{th} task on P_1, and y_i the i^{th} task on P_2. We denote by $l(t_i)$ the execution time (or length) of task t_i.
We represent a schedule by two sets A and B, where A (respectively B) is the set of the tasks scheduled on P_1 (respectively P_2). On each machine, the tasks are scheduled in the decreasing order of execution times. Let $\xi = (A, B)$ be a schedule of the n tasks on the two machines. Let a (respectively b) be a subset of tasks scheduled on P_1 (respectively P_2). We denote by $swap(\xi, a \leftrightarrow b)$ the schedule $((A \setminus a) \cup b, (B \setminus b) \cup a)$. In this new schedule, each machine still executes its tasks using the LPT policy (if two tasks have the same lengths, the one with the smallest identification number is scheduled first).

Let us now consider the following algorithm LPT_{swap}.

Theorem 1 *The algorithm LPT_{swap} achieves an approximation ratio of $\frac{8}{7}$ for the makespan.*

Order tasks by non increasing execution times. At each step i, for $1 \le i \le n$, schedule the current task t_i on the machine which has the smallest completion time. Let LPT denote the schedule obtained in this way.
Let $S = \sum_{i=1}^{n} l(t_i)$.
if $n_2 \ge 2$ *and* $((n_1 = 3$ *and* $l(x_1) + l(x_2) + l(x_3) > (\frac{4}{7})S)$ *or* $(n_1 = 4))$ **then**
  if $n_1 = 3$ **then**
    Let $\xi_1 = swap(LPT, \{x_1\} \leftrightarrow \{y_2\})$, $\xi_2 = swap(LPT, \{x_2\} \leftrightarrow \{y_2\})$ and $\xi_3 = swap(LPT, \{x_1\} \leftrightarrow \{y_1\})$.
    Return a schedule which minimizes the makespan among the schedules: LPT, ξ_1, ξ_2 and ξ_3.
  end
  if $n_1 = 4$ **then**
    Let $\xi_4 = swap(LPT, \{x_3, x_4\} \leftrightarrow \{y_2\})$.
    Return a schedule which minimizes the makespan among the schedules: LPT and ξ_4.
  end
end
else
  Return LPT.
end

Algorithm LPT_{swap}

Proof: Let us assume that ξ is a LPT schedule which does not achieve a $\frac{8}{7}$-approximation ratio. We show that the algorithm LPT_{swap} transforms this schedule into a schedule which achieves a $\frac{8}{7}$-approximation ratio.
Let t_{max} be the last task scheduled on P_1 (the machine with maximum load). We say that a task is *large* if its execution time is greater than or equal to the execution time of t_{max}. A task is *small* if its execution time is smaller than the one of t_{max}.

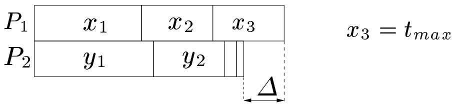

Fig. 1. A LPT schedule.

Claim: There are 3 or 4 tasks on P_1 and exactly 2 large tasks on P_2.

Let us show that we have exactly two large tasks on P_2.

Let us call Δ the difference between the completion time of the last task of P_1 and the last task of P_2 (see Figure 1). The makespan of the schedule ξ is $\frac{\sum_{i=1}^{n} l(t_i) + \Delta}{2}$. Since ξ does not achieve a $\frac{8}{7}$-approximation ratio, we have $\frac{\sum_{i=1}^{n} l(t_i) + \Delta}{2} > \frac{8}{7}OPT$. Since the makespan OPT of an optimal solution is at least $\frac{\sum_{i=1}^{n} l(t_i)}{2}$, we have $\frac{\Delta}{2} > (\frac{8}{7} - 1)OPT$, and so $\Delta > \frac{2}{7}OPT$. Let $\epsilon > 0$ be such that $\Delta = (\frac{2}{7} + \epsilon)OPT$.
We know that $l(t_{max}) \ge \Delta$ (because in a LPT schedule, at each step, we add a

task on the machine with the smallest load), so the minimum execution time of a large task is Δ. We also know that the completion time of the last task on P_2 is smaller than or equal to $OPT - (\frac{1}{7} + \epsilon_1)OPT$, with $\epsilon_1 > 0$, because otherwise the schedule ξ would achieve a $\frac{8}{7}$-approximation ratio. So the maximum number of large tasks on P_2 is equal to $\lfloor \frac{OPT-(\frac{1}{7}+\epsilon_1)OPT}{\Delta} \rfloor = \lfloor \frac{1-(\frac{1}{7}+\epsilon_1)}{\frac{2}{7}+\epsilon} \rfloor = 2$.

Moreover we can deduce that there are at least two large tasks on P_2 because otherwise the schedule ξ would be an optimal schedule.

Let us show now that there are either 3 or 4 tasks on P_1. There are at least 3 tasks on P_1 because otherwise the schedule ξ would be optimal. Indeed, if there is only one task on P_1 it is trivial that ξ is optimal. If there are two tasks on P_1 and $l(x_1) \leq l(y_1)$ then we would not decrease the makespan by putting x_2 and y_1 on the same machine (because P_1 is the machine which has the largest load). If there are two tasks on P_1 and $l(x_1) > l(y_1)$, then $l(y_2) \geq l(x_2)$ (schedule LPT): if we add x_2 to the large tasks of P_2 (y_1 and y_2) we do not decrease the makespan (x_2 starts before the completion time of y_2), and if we add to x_1 one of the large tasks of P_2 (i.e. we exchange x_2 with y_1 or y_2), we do not decrease the makespan because the execution time of each of these tasks (y_1 or y_2) is greater than or equal to the execution time of x_2.

Let us now show that there are at most 4 tasks on P_1.
We know that $l(t_{max}) \geq \Delta > \frac{2}{7}OPT$, and each task on P_1 is larger than or equal to t_{max}. Since the maximum makespan of a LPT schedule on two machines is at most $\frac{7}{6}OPT$ [6] then the maximum number of tasks on P_1 is $\lfloor \frac{(\frac{7}{6})OPT}{\Delta} \rfloor = \lfloor \frac{(\frac{7}{6})OPT}{(\frac{2}{7}+\epsilon)OPT} \rfloor = \lfloor \frac{49}{12} \rfloor = 4$.

We have shown that if a LPT schedule does not achieve a $\frac{8}{7}$-approximation ratio, then it has 3 or 4 tasks on P_1, and at least 2 tasks on P_2. Moreover, $\sum_{t_i \in P_1} l(t_i) > (\frac{8}{7})OPT \geq \frac{8}{7}(\frac{\sum_{i=1}^{n} l(t_i)}{2}) = \frac{4}{7}\sum_{i=1}^{n} l(t_i)$. Thus all the conditions to enter in the first "if" of the algorithm LPT_{swap} are fulfilled. Due to space limitation, the sequel of the proof which proceeds by cases analysis, is omitted here. □

Theorem 2 *The schedule returned by LPT_{swap}, on two machines whose policies are LPT, is a 3-approximate-Nash equilibrium.*

Proof: Let ξ be a LPT schedule of the n tasks and ξ_s be the schedule returned by LPT_{swap}. Let m_ξ denote the makespan of ξ, and m_{ξ_s} the makespan of ξ_s. If $\xi_s = \xi$, then ξ_s is a LPT schedule and it is a Nash equilibrium. Else $\xi_s \neq \xi$, and LPT_{swap} has done a swap: ξ_s is then equal to ξ_1, ξ_2, ξ_3 or ξ_4. Let us show that each task of ξ_s does not have incentive to go on the other machine.

First of all, x_3 and all the small tasks (the tasks whose lengths are smaller than the one of t_{max}) do not have incentive to go on the other machine because if they change they will always be started after at least two large tasks, that is a length greater than $\frac{m_{\xi_s}}{3}$, and without changing they are always completed before or at m_{ξ_s}. Indeed, $l(x_3) \leq \frac{m_{\xi_s}}{3}$ and if a small task (or x_3) changes, it will be scheduled at the earliest at $m_{\xi_s} - l(x_3) \geq m_{\xi_s} - \frac{m_{\xi_s}}{3} > \frac{m_{\xi_s}}{3}$. Likewise, it is trivial that the tasks which are scheduled at the beginning of the schedule (e.g. x_1 and y_1 in ξ_2) do not have incentive to swap.

We then prove by case analysis that every large task of ξ_1, ξ_2, ξ_3 or ξ_4 does not have incentive to change machine. Due to space limitations, the end of the proof is omitted here. □

Corollary 1 *The price of α-approximate stability is at most $\frac{8}{7}$, for all $\alpha \geq 3$.*

Theorem 3 *Let ε be any small constant such that $\varepsilon > 0$. The price of α-approximate stability is at least $\frac{8}{7}$, for all $\alpha \leq 2.1 - \varepsilon$.*

Proof: Let ε be a small number such that $\varepsilon > 0$. Consider the following tasks: a task of length $3.3 - \varepsilon$, a task of length $3 + \varepsilon$, and three tasks of length 2.1. The optimal schedule (for the makespan) of these tasks is achieved if and only if the tasks of length 2.1 are on the same machine, and the other two tasks are on the other machine (see Figure 2 *Left*). The makespan of this schedule is $OPT = 6.3$. In this schedule, the completion time of the task of length $3 + \varepsilon$ is 6.3 but this task could be on the first position if it goes on the other machine (because the policies of the machines are LPT), and its completion time would then be $3 + \varepsilon$. So this schedule is $\frac{6.3}{3+\varepsilon} > (2.1 - \varepsilon)$-Nash approximate.

Fig. 2. *Left:* Optimal schedule for the makespan. This is a $\frac{6.3}{3+\varepsilon}$-approximate Nash equilibrium. *Right:* Approximate schedule for the makespan.

Notice that all the other schedules have a makespan of at least $7.2 + \varepsilon > \frac{8}{7} OPT$. So if we want a schedule which is $\frac{8}{7}$-approximate, this schedule will not be an α-approximate Nash equilibrium, with $\alpha < 2.1 - \epsilon$. Thus the price of α-approximate stability is at least $\frac{8}{7}$, for all $\alpha \leq 2.1 - \varepsilon$. □

An interesting question concerns the relation between the approximation ratio and α, i.e. what happens if we consider other values of the approximation ratio we wish to obtain, or other values of α ? In particular, does there exist algorithms with smaller approximation ratios and which return α-Nash approximate equilibria, with α bounded (and as small as possible)? The following section gives an answer to this question.

3 A variant of Graham's algorithm

We are now interested in $(1+\varepsilon)$-approximate schedules, for any $\varepsilon > 0$. We show that the price of α-approximate stability is smaller than $(1 + \varepsilon)$ if α is at least equal to k, where k is a constant smaller than $\frac{1}{\varepsilon}$.

We consider the algorithm of Graham [6], slightly modified:
(i) Let k be some specified and fixed integer.
(ii) Obtain an optimal schedule for the k longest tasks, such that:

- Once tasks are assigned to each machine, they are scheduled on their machines in order of decreasing lengths (i.e. for a given machine, tasks are scheduled from the largest one to the smallest one).
- If two tasks have the same length, the one which has the smallest identification number is scheduled the first.

(iii) Schedule the remaining $n-k$ tasks using the LPT rule.

This algorithm is a polynomial time approximation scheme (PTAS), and its approximation ratio is $1+\varepsilon$, where ε is equal to $\frac{1}{2+2\lfloor\frac{k}{2}\rfloor}$, if the k largest tasks of the schedule are optimally scheduled [6]. Let us now show that this algorithm, denoted by OPT-LPT(k), returns α-approximate-Nash equilibria, with $\alpha < k-2$.

Theorem 4 *The schedules returned by algorithm OPT-LPT(k) are α-approximate-Nash equilibria, with $\alpha < k-2$.*

Proof: Let us show that each task of an OPT-LPT(k) schedule either does not have incentive to change machine, or does not decrease its completion time by a factor larger than or equal to $k-2$, by going on the other machine. The $n-k$ smallest tasks of the schedule are scheduled using the LPT rule, so they do not have incentive to change machine. Thus we consider the case of the k largest tasks. Let OPT be the optimal solution of these tasks, such as computed by OPT-LPT(k). We now consider three cases.

• In the first case, there are, in OPT, only one task on a machine (w.l.o.g. on P_1), and $k-1$ tasks on the other machine. Since this schedule is an optimal solution, the task on P_1 is necessarily the largest task on the schedule, and this schedule is a LPT schedule. So no task has incentive to change machine in this case.

• Let us now consider the case where there are exactly two tasks on a machine (w.l.o.g. on P_1) in OPT. The others $k-2$ tasks are then on P_2.
We first show that no task scheduled on P_2 has incentive to go on P_1. By construction, we know that $l(x_1)+l(x_2)$ is larger than or equal to the sum of the lengths of the $k-3$ first tasks of P_2, $\sum_{j=1}^{k-3} y_j$. Let i be the largest number such that $l(x_1) \geq \sum_{j=1}^{i} y_j$: the $i+1$ first tasks of P_2 (i.e. the tasks who start at the latest at the end of x_1) do not have incentive to go on P_1, otherwise they would be scheduled after P_1 and would not decrease their completion times. Moreover, we know that $l(x_2) \geq \sum_{j=i+2}^{k-3} y_j$: thus the tasks from y_{i+2} to y_{k-3} do not have incentive to change machine. Likewise, y_{k-2} does not have incentive to change: if it is smaller than x_2, then it would be scheduled on P_1 after x_2, and would not decrease its completion time, since OPT is an optimal solution. If y_{k-2} is larger than x_2, then y_{k-2} starts to be executed before (or at the same time as) x_2. Thus, if it goes on P_1, y_{k-2} will be scheduled after x_1, and then will not

decrease its completion time.
The only task which may have incentive to change machine is x_2. If x_2 is smaller than all the other tasks, then it does not have incentive to change. Otherwise, since OPT is an optimal solution, we know that at least a task of P_2 starts at the same time or after x_2. In the best case, x_2 can go to the first position on P_2: by doing this, it starts on P_2 before at most $k-3$ tasks which started before it when it was on P_1. These $k-3$ tasks are smaller than x_2: the sum of their completion time, S, is thus smaller than $(k-3)\,l(x_2)$. The completion time of x_2 decreases with this change, from $S + l(x_2) < (k-2)\,l(x_2)$ to $l(x_2)$. Thus x_2 is, in OPT, α-Nash-approximate, with $\alpha < k-2$.

• Let us now consider the case where there are exactly $a < k-2$ tasks on P_1, and $b < k-2$ tasks on P_2. Let t be a task on P_1 (respectively P_2) which has incentive to change machine. When it changes machine, t overtakes p tasks of P_2 (respectively P_1), i.e. it starts to be executed before p tasks which started to be executed before t before the change. We know that p is smaller than $k-2$ because there are less than $k-2$ tasks on each machine. Moreover, these tasks have a length smaller than the one of t, otherwise t would not overtake them. Thus, in the best case, t overtakes $k-3$ tasks of length almost equal to $l(t)$, and the completion time of t decreases from a value smaller than $(k-2)\,l(t)$ to $l(t)$. Thus t is α-Nash-approximate, with $\alpha < k-2$. □

Theorem 5 *Let ε be any small number such that $0 < \varepsilon < 1$. OPT-LPT(k) can return α-approximate Nash equilibria, with $\alpha \geq k-2-\varepsilon$ and $k \geq 5$.*

Proof: Let $\varepsilon' = \frac{\varepsilon}{k-2-\varepsilon}$, and let us consider the following instance: a task of length $k-3-\varepsilon'$, a task of length $1+\varepsilon'$ denoted by t, and $k-2$ tasks of length 1. In the only optimal solution for this instance, it can be shown that t is $(k-2-\varepsilon)$-approximate. The schedule returned by OPT-LPT(k) on this instance is an α-approximate Nash equilibrium, with $\alpha \geq k-2-\varepsilon$. □

We can deduce from Theorem 4, and from the fact that the approximation ratio of OPT-LPT(k) is $\frac{1}{2+2\lfloor\frac{k}{2}\rfloor}$, the following result:

Corollary 2 *Let k be any integer larger than or equal to 5. The price of α-approximate stability is at most $1+\varepsilon$, where $\varepsilon = \frac{1}{4+2\lfloor\frac{k}{2}\rfloor} < \frac{1}{k}$, for all $\alpha \geq k$.*

Note that if we want to get an algorithm $\frac{8}{7}$-approximate which returns solutions as stable as possible, then LPT_{swap} is better than OPT-LPT(k): indeed, the solution of LPT_{swap} can be found faster (in OPT-LPT(k) we have 64 schedules to compare, whereas in LPT_{swap} there are at most 4 schedules to compare), and the OPT-LPT(k) returns 4-approximate-Nash equilibria (versus 3-approximate Nash equilibria for LPT_{swap}). On the other hand, OPT-LPT(k) is useful if we wish algorithms with smaller approximation ratios, since OPT-LPT(k) is a PTAS.

4 Tradeoffs

We first show that if we want a price of α-approximate stability smaller than or equal to $(1+\varepsilon)$, then α must be larger than a certain constant in $\Theta(\varepsilon^{-1/2})$.

Theorem 6 *Let $\varepsilon > 0$ and $k > 0$ such that $\varepsilon = \frac{1}{k\,(k+1)}$. Then to get a price of α-approximate stability smaller than $1+\varepsilon$, we have to set $\alpha \geq k$.*

Proof: Consider the following instance: a task of length $k-1$, a task of length 1, and $k+1$ tasks of length $\frac{k}{k+1}$. The optimal schedule (for the makespan) of these tasks is achieved if and only if the tasks of length $\frac{k}{k+1}$ are on the same machine, and the other two tasks are on the other machine (see Figure 3 *Left*). The makespan of this schedule is $OPT = k$. This schedule is a k-approximate Nash equilibrium. Indeed, the completion time of the task of length 1 is k but this task could be on the first position if it goes on the other machine (because the policies of the machines are LPT), and its completion time would then be 1 (which is n times smaller than its current completion time).

Fig. 3. *Left:* Optimal schedule for the makespan. This is a k-approximate Nash equilibrium (the policies of the machines are LPT). *Right:* Approximate schedule for the makespan.

Figure 3 *Right* shows the second smallest makespan schedule with these tasks: all the schedules which are not the optimal one have a makespan greater than or equal to the makespan of this schedule. This makespan is $k - (\frac{k}{k+1}) + 1$. So the ratio between this makespan and OPT is $\frac{k-(\frac{k}{k+1})+1}{k} = 1 + \frac{1}{k} - \frac{1}{k+1} = 1 + \frac{1}{k\,(k+1)}$. Thus if we want a schedule which is $(1+\varepsilon)$-approximate, with $\varepsilon < \frac{1}{k\,(k+1)}$, this schedule will not be an α-approximate Nash equilibrium, with $\alpha < k$. □

We can also prove:

Theorem 7 *The price of α-approximate stability is at least 1.1 (respectively $\frac{9}{8}$), for all $\alpha \leq \frac{10}{3}$ (respectively $\alpha \leq \frac{8}{3}$).*

Figure 4 illustrates the tradeoffs between the approximation ratio of an algorithm, and the stability of the schedules it can returns. The dark grey zone illustrates the results showed in Theorem 6 (for $2 \leq k \leq 10$), Theorem 3, and Theorem 7. Each point (x, α) belonging to the dark grey zone represents a negative result, i.e. it means that there is no x-approximate algorithm which returns α-Nash equilibria. The light grey zone illustrates the results showed by Theorem

4 and Theorem 1: each point (x, α) belonging to the light grey zone represents a positive result, i.e. it means that there is an x-approximate algorithm which returns α-Nash equilibria (this algorithm is either OPT-LPT(k) or LPT_{swap}). If a point (x, α) belongs to the white zone, then it means that we do not have any algorithm corresponding to this point, nor any impossibility result.

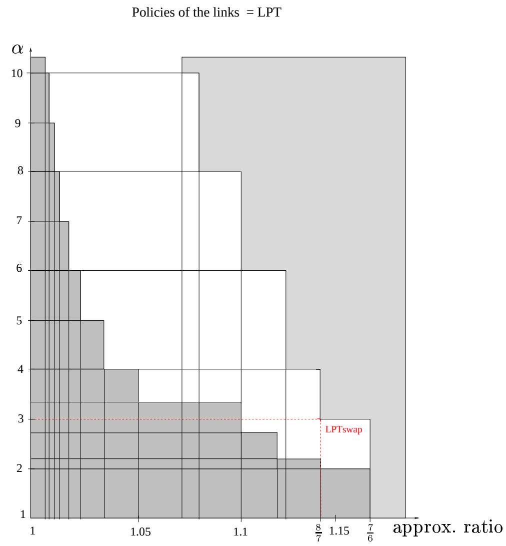

Fig. 4. *Light grey* (respectively *Dark grey*): Relation between α and approximation ratios that it is possible (respectively it is not possible) to obtain if we wish an algorithm which returns α-approximate Nash equilibria.

References

1. G. Christodoulou, E. Koutsoupias, A. Nanavati *Coordination mechanisms.* In Proc. of ICALP 2004, LNCS 3142, pages 345-357.
2. E. Koutsoupias, C. H. Papadimitriou *Worst-case equilibria.* In Proc. of STACS 1999, LNCS 1563, pages 404-413.
3. A. S. Schulz, N. Stier Moses *On the performance of user equilibria in traffic networks.* In Proc. of SODA 2003, pages 86-87.
4. E. Anshelevich, A. Dasgupta, J. Kleinberg, É. Tardos, T. Wexler, T. Roughgarden *The price of stability for network design with fair cost allocation.* In Proc. of FOCS 2004, pages 295-304.
5. R. Lipton, E. Markakis, A. Mehta *Playing Large Gamed Using Simple Strategies.* ACM Conference on Electronic Commerce, pp. 36-41, 2003.
6. R. Graham *Bounds on Multiprocessing Timing Anomalies.* SIAM Jr. on Appl. Math., 17(2), pp.416-429, 1969.
7. D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, P. Spirakis, *The structure and complexity of nash equilibria for a selfish routing game.* In Proc. of ICALP 2002, LNCS 2380, pages 123-134.