Equilibria in Online Games[*]

Roee Engelberg[†] Joseph (Seffi) Naor[‡]

Abstract

We initiate the study of scenarios that combine online decision making with interaction between non-cooperative agents. To this end we introduce *online games* that model such scenarios as non-cooperative games, and lay the foundations for studying this model. Roughly speaking, an online game captures systems in which independent agents serve requests in a common environment. The requests arrive in an online fashion and each is designated to be served by a different agent. The cost incurred by serving a request is paid for by the serving agent, and naturally, the agents seek to minimize the total cost they pay. Since the agents are independent, it is unlikely that some central authority can enforce a policy or an algorithm (centralized or distributed) on them, and thus, the agents can be viewed as selfish players in a non-cooperative game. In this game, the players have to choose as a strategy an online algorithm according to which requests are served. To further facilitate the game theoretic approach, we suggest the measure of *competitive analysis* as the players' decision criterion.

As the expected result of non-cooperative games is an equilibrium, the question of finding the equilibria of a game is of central importance, and thus, it is the central issue we concentrate on in this paper. We study some natural examples for online games; in order to obtain general insights and develop generic techniques, we present an abstract model for the study of online games generalizing *metrical task systems*. We suggest a method for constructing equilibria in this model and further devise techniques for implementing it.

1 Introduction

Consider a scenario in which several processes share a common cache according to the following rules. Over time, the processes receive requests to access pages, where each page request is designated to be accessed by a specific process (note that as the accessed data is also shared, different processes can request the same page). Each time a process is requested to access a page, the page must be cached first. If the cache is full, the process that needs to access the page decides which page will be evicted from the cache. Naturally, the processes seek to minimize the number of page faults incurred by their requests. Assuming that the processes are independent, they are expected to exhibit selfish behavior. Hence, each process strives to use as his paging policy an online algorithm which is the *best response* to the policies used by the other processes. In what respect should this response be the best? Equivalently, according to which decision criterion should the response be chosen? As the requests arrive in an online manner, and since the processes have no prior information about the requests (e.g. requests distribution), we concentrate on competitiveness, which by now is the standard measure used for evaluating online algorithms, as the measure for the performance of

[*]A preliminary version appeared in the Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA (2007), pp. 149–158.

[†]Computer Science Department, Technion, Haifa 32000, Israel. E-mail: *roee@cs.technion.ac.il*

[‡]Computer Science Department, Technion, Haifa 32000, Israel. E-mail: *naor@cs.technion.ac.il*

possible paging policies. That is, a process chooses the paging policy with the best competitive ratio, *given the policies of the other processes*.

Suppose, for example, that all the processes are advised to use a *least recently used* ($\mathcal{LRU}$) policy when evicting pages from the cache. Can any process guarantee itself a better competitive ratio by choosing a different (deterministic) policy? In other words, does the $\mathcal{LRU}$ policy constitute an equilibrium? Such questions come up in the context of many other scenarios that share similar attributes. However, it appears that previous work has either concentrated on the inherent uncertainty that comes from the online nature of the problem, or on game-theoretic aspects of selfish agents operating in a distributed system. In this paper we develop a formal model for such scenarios in order to initiate the study of the issues just raised.

At first, consider a general setting in which there are several independent agents operating jointly in a common environment. The environment is likely to include some common infrastructure or resources that the agents may use, but as the agents are not necessarily identical, they may have different requirements or different access to the given resources. The agents need to satisfy a sequence of tasks, where each task is allocated to a specific agent. The tasks are given in an online fashion; the agents need to serve the tasks upon arrival, by making decisions based on the current state of the system, without knowing future events (specifically, there is no known distribution on the possible requests). Each agent runs, *independently*, an algorithm minimizing the cost it pays for serving the tasks.

Traditionally, such scenarios are modelled as an *online problem*, where the goal is to design an algorithm that minimizes the total cost incurred by the actions of *all* the agents. Essential to modelling the above as an online problem is the assumption that all the agents are controlled and managed by the same authority. However, in general, this assumption is problematic as the agents are independent, and hence, it is not likely that there exists a central authority that can enforce a policy or an algorithm (centralized or distributed) on the agents. Furthermore, the independence between the agents means that each agent is concerned about improving its own performance, while being indifferent to the performance of other agents. This immediately implies a possible lack of coordination between the agents, leading to selfish behavior, implying that an agent might deviate from a given algorithm if such a deviation improves its performance.

Such a deviation is crucial in many cases. For example, when designing protocols to be used as standards, no matter how efficient a protocol may be, it cannot be used as a standard if users have an incentive to deviate from it. Hence, a protocol used as a standard should be designed to induce an equilibrium among its users.

We conclude that modelling systems as a centralized online problem fails to capture a fundamental attribute, thus motivating the use of a game theoretic approach that takes into consideration the lack of coordination between agents. Such an approach can be used to predict the behavior of agents in a system, as well as to design stable standards that define equilibria. Recently, there has been a growing literature at the interface of game theory and algorithms, and in particular concerning the stability of standards (e.g. [1, 9, 17]). Nevertheless, so far, scenarios having an online nature were not addressed using that approach.[1]

1.1 Online Games

We now explain how to model a system as an *online game*. Basically, an online game is a *pre-Bayesian game* (equivalently, a *game in informational form* or an *incomplete information game with strict type of uncertainty*; for example, see [12]). In what follows we informally define the relevant components of the game: the players, their strategy spaces and cost (utility) functions, and possible states.

[1]Some works ([6, 10, 11, 14, 15, 16]) combine online decision making with mechanism design, but the online decision-making role is limited to the mechanism itself, while the role of the agents is merely to supply the input to the online algorithm.

In an online game there are N players denoted by $\{1, \ldots, N\}$. As each player controls a single agent in the system, the strategy space of player i is the set of legal (defined below) online algorithms that determine the actions of the ith agent. Each possible state consists of a request sequence. The requests arrive in an online fashion into the system, and the agents run the online algorithms chosen by the players. Each request is allocated to a specific agent, and thus an online algorithm for a player is considered legal if, for every request sequence, each request allocated to a player's agent is served before the next request to the system arrives. Each player is charged for the cost of all the operations of its agent.

The stable outcomes of the interactions of non-cooperative selfish agents correspond to the *equilibria* of the underlying game, that is, the points where unilateral deviation does not help any user improve its performance. An equilibrium of an online game is a vector of strategies, one for each player, from which no player has an incentive to deviate. While in strategic games the incentive for deviation is simply a lower cost, this is not the case in pre-Bayesian games due to the inherent uncertainty regarding the realized state. Moreover, as the players have no prior information about the realized state, the definition of a best response strategy traditionally relies on decision criteria that stem from worst case considerations (see [12, 2]). Some examples for such decision criteria are minimizing the maximum cost (taken over all possible states), minimizing the maximum regret (the difference between the cost paid and the minimum possible cost for the same state), and minimizing the competitive ratio.

Accordingly, the best response in online games must also be defined primarily according to worst case decision criteria. We emphasize that this is a straight-forward implication of the inherent uncertainty in online games, and there does not seem to be any other reasonable alternative definition avoiding worst case considerations. Nevertheless, one might consider further refinement of the best response definition in which some additional criteria characterize the better strategy among the strategies having the same performance with respect to a primary worst case measure.

We focus on the competitive ratio as the decision criterion. Roughly speaking, it means that players measure the performance of their strategy with respect to an optimal offline strategy determined by an adversary that "knows" in advance both the request sequence and the players' strategies. Specifically, *given the strategies of the other players*, a player minimizes the worst case ratio (taken over all possible request sequences) between the cost it pays and the cost paid by the adversary. This ratio is called the *competitive ratio*, although notice that a strategy might have different competitive ratios with respect to different strategies of the other players.

Accordingly, we study the competitive ratio equilibrium, which is a vector of strategies, one for each player, in which no player can achieve a better competitive ratio by changing its strategy. As the best response strategy might be hard to design, implement, or compute, the equilibrium notion is relaxed, and we also consider *approximate* equilibria in which every player achieves a competitive ratio that is within a known factor away from the best competitive ratio attainable by any strategy. Identifying the set of solutions of an online game which are in (approximate) equilibrium is at the heart of the analysis of the game we define, and comprises the basis for performance evaluation in our system.

We note that the players in an online game may not necessarily "know" what are the strategies of the other players, however, as in a Nash equilibrium of any strategic game, the strategy of each player is a best response to the other players' strategies. Moreover, as each strategy is actually an online algorithm, such a strategy can capture different responses to different events that might occur while the players are serving the request sequence.

1.2 Examples

We now turn to special cases that demonstrate the general settings discussed above. There are many scenarios that involve interaction between different agents: processors in a distributed system; processes in a computer; users accessing a server on the network; routers in a network; users in a peer-to-peer systems; and many more. For clarity, we concentrate on scenarios that were already modelled and studied as (centralized) online problems, yet the inherent selfishness of the users in these scenarios was not taken into account. Thus, the concept of an online game better models these examples.

We have already seen the *paging game* that generalizes the classic online paging problem. As a second example we consider *file caching* in distributed systems, e.g., peer-to-peer caches and web caches. In this caching game, there are N servers and each controls its own cache. The servers get requests for files and should serve them either by caching the file to their own cache or by accessing a remote replica of the file that is cached by another server. Caching a file incurs some (possibly constant) cost and might be constrained by the capacity of the cache, while accessing a remote replica incurs a cost which is typically proportional to the distance between the accessing server and the location of the replica.

Such scenarios have been previously examined and analyzed using different approaches. Early studies modelled the above scenario as an online problem (e.g., the *distributed paging* problem studied by Bartal, Fiat and Rabani in [4]). As can be expected, such models tend to ignore the selfish behavior of the servers in distributed systems. Recent works (e.g. [8]) indeed use a game-theoretic approach to overcome this problem, but they model caching scenarios as offline problems, while most of these scenarios are inherently online. We argue that the online caching game better describes these scenarios as it combines the advantages of the above approaches and captures the essential properties of such systems - both selfishness and online nature.

We consider the *online generalized Steiner game* where there are ISP's operating in a common environment (which includes potential users) modelled as an edge-weighted undirected graph. Requests for connecting pairs of nodes in the graph arrive online. A request is served when all the edges of a path between the given nodes belong to the solution. Each request is designated to a specific ISP, that serves the request and pays for the edges that it adds to the solution. An edge which is added to the solution can be re-used without being paid for again.

As these examples indicate, many scenarios have both an online nature and interaction between different agents, and thus are better described as online games. We thus turn to formally discuss the model of online games in its full generality, so as to establish the foundations for studying these scenarios. This discussion is followed by the study of specific online games that arise in many contexts.

1.3 Our Results

We present the new model of online games, which is used to study scenarios that are intrinsically online and in which different agents interact. We suggest the measure of competitive analysis as the players' decision criterion, and introduce the appropriate basic concepts of the model, including the equilibria and the approximate equilibria of the game. We point out the question of finding equilibria as being of central importance. We also pay special attention to the incorporation of randomness in the model, as it differs from the theory of classic strategic games. We further introduce two desired properties of equilibria - *efficiency* and *sub-game perfect equilibrium*. Informally, the efficiency property excludes equilibria in which the players use "unfriendly" strategies that prevent the achievements of a good competitive ratio. Sub-game perfect equilibria are more stable than other equilibria.

In order to obtain general insights and develop generic techniques, we present an abstract model for the study of online games - the *metrical task game*. This model generalizes *metrical task systems*, a well-known abstract model for studying online problems. We identify a general useful principle for deriving equilibria in metrical task games, the *non-malleability principle*, which, roughly speaking, suggests that if the interaction between the agents is nullified, an equilibrium is easier to derive. We further develop the *optimistic method* that is aimed to implement the non-malleability principle by having each player assume a certain behavior on the part of the other players. Relying on that method, we devise techniques to construct equilibria (or even sub-game perfect equilibria) in some classes of metrical task games. According to these techniques, given an online game, some induced online problems are examined and competitive algorithms for these problems are designed and further exploited for constructing the equilibrium. The use of these techniques is demonstrated as they are applied in our study of some online games, e.g. the online generalized Steiner game.

Our discussion is followed by the study of a few natural examples for online games, some of which were introduced earlier. For the paging game we show that $\mathcal{LRU}$ defines an efficient sub-game perfect equilibrium with respect to deterministic strategies, while "plain" $\mathcal{FIFO}$ is not even competitive with respect to the individual player. Interestingly, this result gives another justification for preferring $\mathcal{LRU}$ over $\mathcal{FIFO}$. For the caching game we design a strategy which yields an approximate equilibrium with respect to deterministic strategies, while for the online generalized Steiner game we show how to exploit the algorithm of [5] for the corresponding online problem to construct an efficient approximate equilibrium of the online game.

As this paper introduces the new concept of online games, we establish the foundations needed for the study of this new area of research. While doing so, the study of online games is far from being complete. Thus, many questions are left open, and we note some possible further research directions.

1.4 Organization

The rest of this paper is organized as follows. In Section 2 we formally define the central terms and notions used in the study of the online games. In Section 3 we study the paging game, while in Section 4 we study the list accessing game. Section 5 introduces the abstract model of metrical task games, and some general techniques for constructing an equilibrium. These techniques are used in Section 6 where the file caching game is discussed and in Section 7 where we study the online generalized Steiner game. We conclude in Section 8 with open questions and future research directions.

2 Preliminaries

We now turn to formalize the basic notions of online games. We note that the following definitions are stated with respect to cost minimization online games only, while the analogous definitions for profit maximization online games can be easily obtained.

We refer the reader to [7, Chapter 7] for the formal definition of an online problem. This definition can be generalized to formally describe the model of online games in the follwing way. Consider an online game with N players. Instead of one set of feasible requests and actions, we are given a set of feasible requests and actions for each player. A strategy for a player is an online algorithm, that is, a function that assigns an action for each feasible history that ends with a request that is designated for that player. Notice that here the difference between an online game and an online problem is that a history in an online game includes additional information about the identity of the players for each request/action. Accordingly, when an online game generalizes a known online problem, we can use known online algorithms as strategies in a *straight forward* manner, that is, given the function from histories to actions that describes the algorithm, the corresponding strategy is a simple generalization that ignores the additional information about the players given in the history.

We use the standard notation for the strategy vector of the players, i.e. we denote by $s = (s_1, s_2, \ldots, s_N)$ a strategy vector where s_i is the strategy (the online algorithm) played by the ith player. We denote by s_{-i} the vector of strategies played by all the players except for the ith player. Given a request sequence $\sigma = r_1, r_2, \ldots, r_m$, let $pref_j(\sigma)$ be the prefix of σ of length j, i.e. $pref_j(\sigma) = r_1, r_2, \ldots, r_j$. We denote by $p(r_i)$ the player that serves request r_i.

Given an online deterministic algorithm $\mathcal{A}$ for the ith player, we use $\mathcal{A}_{s_{-i}}(\sigma)$ to denote the cost incurred by $\mathcal{A}$ when serving the ith player requests in the request sequence σ, while other players play the strategies given by s_{-i}. We denote by $\mathcal{OPT}$ an optimal (offline) strategy for the ith player, i.e. a strategy that while serving the ith player's requests incurs the minimum possible cost. We note that by this definition $\mathcal{OPT}$ is assumed to know in advance both the entire request sequence σ and the strategy vector s_{-i}.

Definition 1. *A deterministic algorithm $\mathcal{A}$ is said to be c-*competitive for the ith player with respect to a strategy vector s_{-i} *if there exists an α such that for every request sequence σ, $\mathcal{A}_{s_{-i}}(\sigma) \leq c \cdot \mathcal{OPT}_{s_{-i}}(\sigma) + \alpha$. The infimum over the set of all values c such that $\mathcal{A}$ is c-competitive for the ith player with respect to the strategy vector s_{-i} is called the* competitive ratio of $\mathcal{A}$ for the ith player with respect to s_{-i}*, and is denoted by $\mathcal{R}_{s_{-i}}(\mathcal{A})$.*

In many of the scenarios modelled as online games the players are almost identical. Naturally, in such cases, the players will tend to use the same strategy in equilibrium, especially when such a strategy is agreed upon as a *standard*. Moreover, in cases of inherent symmetry between the players, e.g., symmetric games, it is easier to design equilibria that consist of one identical strategy, motivating the following definition.

Definition 2 (Self-Competitiveness). *A strategy $\mathcal{A}$ is said to be c-*self-competitive *if for every j it is c-competitive for the jth player with respect to the strategy vector $s_{-j} = (\mathcal{A}, \ldots, \mathcal{A})$.*

When considering deterministic strategies in an online game, a *competitive ratio equilibrium* (or simply *an equilibrium*) is a strategy vector $s = (s_1, \ldots, s_N)$ such that for every i, and for every possible deterministic strategy s'_i, $\mathcal{R}_{s_{-i}}(s_i) \leq \mathcal{R}_{s_{-i}}(s'_i)$. As equilibria are sometimes hard to construct or compute, the notion of *approximate* equilibria is also considered. A strategy vector s (of deterministic strategies) is an *α-equilibrium with respect to deterministic strategies*, if for every i and every possible deterministic strategy s'_i, $\mathcal{R}_{s_{-i}}(s_i) \leq \alpha \cdot \mathcal{R}_{s_{-i}}(s'_i)$. We note that, by definition, every c-self-competitive strategy $\mathcal{A}$ gives rise to the c-equilibrium $(\mathcal{A}, \ldots, \mathcal{A})$. Moreover, for a c-self-competitive strategy $\mathcal{A}$, the vector $(\mathcal{A}, \ldots, \mathcal{A})$ might even be c'-equilibrium for $c' < c$. To prove such a result, one should prove a lower bound on $\min_{\mathcal{B}} \mathcal{R}_{s_{-i}}(\mathcal{B})$ where $s_{-i} = (\mathcal{A}, \ldots, \mathcal{A})$.

We now elaborate on several desired properties that equilibria solutions should satisfy.

Efficiency. In some games, an equilibrium can be constructed using strategies in which players mutually prevent the achievement of a good competitive ratio. Such equilibria are unlikely to get realized, as players tend to have very poor performance in these situations. This motivates defining measures for evaluating the efficiency of an equilibrium in an online game. We use the competitive ratio of each of the players in a given equilibrium as a measure that reflects the individual efficiency of each of the players. We also evaluate the overall efficiency of the system, including all agents, as follows. Consider a strategy vector as an online algorithm for the "social" online problem defined by the online game when all agents are controlled by a single entity. Let the competitive ratio of the strategy vector be its competitive ratio as an online algorithm for the appropriate "social" online problem. We use this competitive ratio as a measure for the overall efficiency of equilibria. Note that an optimal (competitiveness-wise) online algorithm that has centralized control of all the agents can be referred to as a *social optimum* of the online game, and accordingly, the closer the competitive ratio of an equilibrium to that of the social optimum, the better that equilibrium is. This motivates the study of the *price of stability* in online games, a topic that is deferred for further research.

Sub-game Perfect Equilibrium. The continual nature of online games implies that players might rethink their strategy as they serve the request sequence. Given a strategy vector defining an equilibrium, the players have no incentive to change their strategy while serving the requests, as long as no deviations from the given strategy vector happen[2]. However, in case a deviation occurs, there may be an incentive to one (or more) of the players to further deviate, and this process may justify itself, even though it probably should not have happened in the first place. Hence, an equilibrium is considered to be more stable if it is resistent to some "local" deviations. Equilibria possessing this property are called *sub-game perfect equilibria*. Formally, a *history* of an online game G is a legal sequence of requests and reactions of the players. Accordingly, every history h induces a *sub-game* of G which is the online game defined according to G with the history h being initially realized. Finally, a *sub-game perfect equilibrium of an online game* G is a vector of strategies s such that for every sub-game G' of G it holds that the strategy vector induced by s in G' is an equilibrium of G'.

2.1 Randomized Strategies

One of the differences between strategic games in the normal form and pre-Bayesian games (including online games) is the implications of randomization. A *mixed strategy of player* i in a strategic game (or a *randomized strategy*) is a probability distribution over the deterministic strategies of player i. The same definition holds for pre-Bayesian games, and a mixed strategy in an online game is what we can intuitively interpret as a randomized online algorithm (strategy). Allowing mixed strategies in an online game gives rise to a new online game, which is referred to as the *mixed extension* of the given online game. To complete the definition of the *mixed extension* of an online game, note that given a vector of mixed strategies and a request sequence, the cost that a player pays is the expected cost, where the expectation is computed according to the given distributions over the strategies.

As the analysis of the mixed extension of an online game should comply with the decision criterion of competitive analysis, we extend the definition of competitive ratio accordingly. Notice that due to the worst case attribute of this decision criterion, the competitive ratio of a mixed strategy is *not* the expectation of the competitive ratios of the underlying deterministic strategies. To better understand this subtle issue, one can think of competitive analysis of randomized online algorithms as the concept we apply. As the competitive ratio of randomized online algorithms is determined with respect to a specific kind of *adversary*, we have a different set of definitions for each type of adversary. Currently, we focus on oblivious adversaries.

Definition 3. *A randomized algorithm $\mathcal{A}$ is said to be c-*competitive *for the i*th player *with respect to a* strategy vector s_{-i} *(which can include randomized strategies), if there exists α such that for every request sequence σ, $E_r[\mathcal{A}_{s_{-i}}(\sigma)] \leq c \cdot E_r[\mathcal{OPT}_{s_{-i}}(\sigma)] + \alpha$, where r denotes the coin tosses of all the players.*

[2]In fact, one also needs to assume that the strategies of the players are optimized with respect to every prefix - otherwise, given a non-worst case prefix, a player might have an incentive to change its strategy. We note that in most online games there are no non-worst case prefixes (and hence such incentives are irrelevant), and moreover, the notion of sub-game perfect equilibria captures stability even in the presence of such prefixes.

Here, assuming an oblivious adversary, $\mathcal{OPT}$ does not know in advance the coin tosses of the players, but rather only knows their actions (responses) till the time it has to serve its requests. Now, the *competitive ratio of $\mathcal{A}$ for the ith player with respect to s_{-i}* (oblivious adversary), can be defined similarly to Definition 1, and hence denoted by $\mathcal{R}_{s_{-i}}(\mathcal{A})$. The definition of self-competitiveness follows similarly to Definition 2.

While the mixed extension of a strategic game preserves some of the properties of the original game (played without randomization), extending an online game (and in general, a pre-Bayesian game) to allow randomized strategies does not have this property. Particularly, one of the important properties is that in a strategic game, every Nash equilibrium of the game is also a Nash equilibrium of its mixed extension. This is not the case for online games, due to the worst case attribute of the decision criterion used by the players. Hence, when considering both deterministic strategies and randomized strategies, the appropriate formal definitions of *equilibrium* and *approximate equilibrium* of an online game follow from the definitions with respect to the mixed extension.

3 The Paging Game

In the paging game a cache of size k and a set of pages are shared between N processes. The processes get requests for pages, and each page has to be retrieved to the cache before being accessed by the processes. When a process needs to cache a page and the cache is full, it should choose which page to evict from the cache. If request r_i incurs a page fault, player $p(r_i)$ is charged for the page fault. In what follows we prove the interesting difference between the known paging policies $\mathcal{LRU}$ and $\mathcal{FIFO}$: while the former constructs an efficient sub-game perfect equilibrium, the latter results in a strategy vector in which the players have an unbounded competitive ratio. Notice that in general, the known lower bounds for the online paging problem are still valid here, in particular the lower bound of k on the competitiveness of any deterministic online algorithm, and the lower bound of H_k on the competitiveness of any randomized online algorithm (see [7]).

3.1 Algorithm LRU

Algorithm $\mathcal{LRU}$ for online paging is known to have competitive ratio of k (see [7]). A player in the paging game can use $\mathcal{LRU}$ as a strategy in one of two possible ways. The first, referred to as $\mathcal{LRU}_{self}$, is to ignore the requests served by the other players, and in case of a page fault to evict its least recently used page. The second way is by taking into consideration all requests, including those served by other players. Notice that this is the straight forward generalization of the online algorithm, hence we denote this strategy by $\mathcal{LRU}$, and note that it requires that the players know the time of the recent accesses to the pages in the cache. Although one might expect that $\mathcal{LRU}_{self}$ is a better strategy than $\mathcal{LRU}$ for the selfish player, this is not the case: $\mathcal{LRU}_{self}$ is not c-self-competitive for every c, while $\mathcal{LRU}$ is k-self-competitive. This can be easily understood by recalling that the accessed pages are shared among all the processes. Hence, no process has a set of pages that might be considered as its "own" pages and which it prefers to keep in the cache.

Lemma 1. *$\mathcal{LRU}_{self}$ is not c-self-competitive for every c.*

Proof. For simplicity, assume that there are two players, $k = 2$, and that there are 3 different pages, and consider the following (arbitrary long) request sequence: $r_1 = 1$, $p(r_1) = 1$; $r_2 = 1$, $p(r_2) = 2$; $r_3 = 2$, $p(r_3) = 1$; $r_4 = 2$, $p(r_4) = 2$; $\forall i > 2$, $r_{2i-1} = 3$, $p(r_{2i-1}) = 1$ and $r_{2i} = 1$, $p(r_{2i}) = 2$. If both players use $\mathcal{LRU}_{self}$ as their strategy, there will be a page fault in every request starting from the 5th request. A better strategy for the first player would be to evict page 2 when it serves the 5th request, and this would incur no further page faults. □

Theorem 2. *$\mathcal{LRU}$ is k-self-competitive.*

Proof. We use the notion of k-phase partition (see [7, p. 37]). Fix a user i and a request sequence σ, and let $s_{-i} = (\mathcal{LRU}, \dots, \mathcal{LRU})$. Given a strategy $\mathcal{ALG}$ for the ith player, we denote by $S_j(\mathcal{ALG})$ the set of pages in the cache immediately after the first $j-1$ requests were served and before the jth request is revealed.

We divide the request sequence into *charging phases*. Each charging phase is a non-empty subsequence of σ. The first phase starts at the first request, while the other phases start right after the previous phase ends; all the phases, except maybe the last phase, end one request before the *later* of the following two events happens: There are requests for $k+1$ different pages since the beginning of the phase; $\mathcal{OPT}$ evicts a page from the cache as a response to a request that is not the first request in the phase.

We now show that in each charging phase, $\mathcal{LRU}$ will pay at most k. Fix a phase ρ. If the phase does not terminate according to the second condition, then the claim follows from the definition of $\mathcal{LRU}$ (after a page is requested for the first time in ρ, it will not be evicted during ρ). Otherwise, let r_j be the first request for the $(k+1)$st different page in ρ. As before, $\mathcal{LRU}$ will pay at most k until r_j. Then, $S_{j+1}(\mathcal{LRU}) = S_{j+1}(\mathcal{OPT})$, as both caches will include all the k most recently used pages (as $\mathcal{OPT}$ does not evict pages and all the other players use $\mathcal{LRU}$ as their strategy). Now, as long as the second condition does not hold, both caches will contain the same set of pages. Obviously, if $S_l(\mathcal{LRU}) = S_l(\mathcal{OPT})$, and r_l results in a page fault for $\mathcal{LRU}$, then r_l results in a page fault for $\mathcal{OPT}$ too. Thus, the above implies that starting at r_j, $\mathcal{LRU}$ will not evict any page till ρ ends.

The second termination condition for a phase implies that for every phase, except, perhaps, for the last one, we can match a different page eviction performed by $\mathcal{OPT}$, and the theorem follows. $\square$

We conclude that $(\mathcal{LRU}, \dots, \mathcal{LRU})$ is an equilibrium with respect to deterministic strategies. Notice it is optimal (with respect to deterministic strategies) according to both our efficiency criteria: the competitive ratio of the players is the lowest possible, and so is the competitive ratio of the equilibrium itself, as the interaction between the strategies results in the same behavior of the online (centralized) $\mathcal{LRU}$ paging policy. Moreover, the proof of Theorem 2 can be generalized to prove that this equilibrium is in fact a sub-game perfect equilibrium.

3.2 Algorithm FIFO

Algorithm $\mathcal{FIFO}$ for online paging is k-competitive [7]. To derive an appropriate strategy for the paging game from $\mathcal{FIFO}$, assume that each page in the cache has a tag specifying the last time it was brought to the cache. A player that uses the $\mathcal{FIFO}$ strategy evicts the oldest page in cache.

Theorem 3. $\mathcal{R}_{(\mathcal{FIFO},\dots,\mathcal{FIFO})}(\mathcal{FIFO}) = \infty$.

Proof. We construct an arbitrarily long sequence of requests such that the cost paid by the first player when using $\mathcal{FIFO}$ is arbitrarily large compared to the cost it pays when using a different strategy.

For convenience, in what follows we assume that k is even. Assume that initially the cache is empty, and that there are $\frac{3}{2}k$ different pages which will be denoted by $\{1, \dots, \frac{3}{2}k\}$. The request sequence is defined by:

$$r_i = \begin{cases} i & i = 1, \dots, k+1 \\ j & i = k+2j,\ k > j \geq 1 \\ r_{i-(2k-1)} & i = k+2j,\ j \geq k \\ k+j+1 & i = k+1+2j,\ \frac{k}{2} > j \geq 1 \\ r_{i-(k+1)} & i = k+1+2j,\ j \geq \frac{k}{2} \end{cases}$$

and

$$p(r_i) = \begin{cases} 2 & i = 1, \ldots, k \\ 1 & i = k+1 \\ 1 & i = k+2j,\ j \geq 1 \\ 2 & i = k+1+2j,\ j \geq 1 \end{cases}.$$

The following tables sketch the different phases of the sequence.

	Initial phase				
i	1	2	$\ldots$	k	$k+1$
r_i	1	2	$\ldots$	k	$k+1$
$p(r_i)$	2	2	$\ldots$	2	1

	First intermediate phase							
i	$k+2$	$k+3$	$k+4$	$k+5$	$\ldots$	$2k-2$	$2k-1$	$2k$
r_i	1	$k+2$	2	$k+3$	$\ldots$	$\frac{k}{2}-1$	$\frac{3k}{2}$	$\frac{k}{2}$
$p(r_i)$	1	2	1	2	$\ldots$	1	2	1

	Second intermediate phase							
i	$2k+1$	$2k+2$	$2k+3$	$2k+4$	$2k+5$	$\ldots$	$3k-2$	$3k-1$
$i-(k+1)$	k		$k+2$		$k+4$	$\ldots$		$2k-2$
r_i	k	$\frac{k}{2}+1$	1	$\frac{k}{2}+2$	2	$\ldots$	$k-1$	$\frac{k}{2}-1$
$p(r_i)$	2	1	2	1	2	$\ldots$	1	2

	The first part of the final phase				
i	$3k$	$3k+1$	$3k+2$	$3k+3$	$\ldots$
$i-(k+1)$		$2k$		$2k+2$	$\ldots$
$i-(2k-1)$	$k+1$		$k+3$		$\ldots$
r_i	$k+1$	$\frac{k}{2}$	$k+2$	$\frac{k}{2}+1$	$\ldots$
$p(r_i)$	1	2	1	2	$\ldots$

Lemma 4. *If all the players are using the $\mathcal{FIFO}$ strategy, after the kth request and till the end of the sequence, the queue contains the last k requested pages according to the order they have been requested. Specifically, every $k+1$ consequent requests are for different pages and r_i is not in the cache when it is invoked.*

Proof. By induction. Obviously, after the kth request is served, the queue contains the pages $1, \ldots, k$ according to the order they have been requested. Hence $r_{k+1} = k+1$ is not in the cache.

Assume that the claim holds after r_{i-1} is served. There are four different cases.

- $i = k+2j,\ k > j \geq 1$ - as the previous request for r_i is r_j, it follows from the induction hypothesis that r_i is not in the cache.
- $i = k+1+2j,\ \frac{k}{2} > j \geq 1$ - as r_i is the first request for this page, it is obviously not in the cache.
- $i = k+1+2j,\ j \geq \frac{k}{2}$ - by the induction hypothesis, all the pages in the set $\{r_{i-j}\}_{j=1}^{k+1}$ are different. Moreover, the cache includes $\{r_{i-j}\}_{j=1}^{k}$ when r_i is invoked, and hence the claim follows.

- $i = k + 2j,\ j \geq k$ - we need to show that $r_i \notin \{r_{i-j}\}_{j=1}^{k}$. By definition it is enough to prove that $r_{i-(2k-1)} \notin \{r_{i-j}\}_{j=1}^{k}$. Firstly, notice that $r_{i-(2k-1)} \in \{r_{i-k+1-j}\}_{j=-1}^{k-2}$, and by the induction hypothesis
$$r_i = r_{i-(2k-1)} \notin \{r_{i-k-j-1)}\}_{j=-1}^{k-3} . \tag{1}$$
Specifically, it means that $r_i = r_{i-(2k-1)} \neq r_{i-k}$. Similarly,
$$r_i = r_{i-(2k-1)} \notin \{r_{i-j-(2k-1))}\}_{j=1}^{k-1} . \tag{2}$$
Next, as $i \geq 3k$, it holds that $i - (k-1) \geq 2k+1$ and hence $r_{i-j} = r_{i-j-(k+1)}$ for every odd $1 \leq j \leq (k-1)$. Thus, for $j = k-1$ we have that $r_{i-(k-1)} = r_{i-(k-1)-(k+1)} = r_{i-2k}$ and, by (2), $r_i \neq r_{i-2k} = r_{i-(k-1)}$. For every odd $1 \leq j \leq (k-3)$ we have that by (1), $r_i \neq r_{i-k-j-1} = r_{i-j}$.

  It is left to prove that $r_i = r_{i-(2k-1)} \neq r_{i-j}$ for even $2 \leq j \leq k-2$. We distinguish between two case:

  1. $i - j \geq 3k$ - it holds that $r_{i-j} = r_{i-j-(2k-1)}$, and by (2), $r_i \neq r_{i-j-(2k-1)} = r_{i-j}$.
  2. $i - j < 3k$ - it holds that $3k \leq i < 3k + j$ and $k + 1 \leq i - (2k-1) < 3k + j - 2k + 1 = k + j + 1 \leq 2k - 1$ hence $r_i = r_{i-(2k-1)} \geq k+1$. At the same time, since $2k+2 \leq i-j < 3k$, $r_{i-j} = \frac{i-j-k}{2} < k$ and the claim follows.

□

Lemma 5. *If the first player evicts page k when serving r_{k+1}, then he has no further page faults.*

Proof. Assuming that the first player evicts page k when serving r_{k+1}, the $\mathcal{FIFO}$ queue contains $1, \dots, k-1, k+1$ in this order (1 is the next to be taken out) after r_{k+1} is served. Obviously, the next $\frac{k}{2} - 1$ requests that are served by the second player are for pages that are not in the cache (and won't be in the cache when they are requested), and hence the queue is updated accordingly. However, for $1 \leq j \leq \frac{k}{2}$ the first player needs to access page j at time $k + 2j$. Since till this time there has been only $j - 1$ page faults of the second player, and according to the order of the pages in the queue, it follows that r_{k+2j} will not incur a page fault. Accordingly, after r_{2k} is served, the pages $\frac{k}{2}, \frac{k}{2}+1, \dots, k-1, k+1, k+2, \dots, \frac{3k}{2}$ are in the queue in this order.

This is the beginning of the second intermediate phase. Again, the next $\frac{k}{2}$ requests that are served by the second player $(k, 1, 2, \dots, \frac{k}{2} - 1)$ are for pages that are not in the cache (and won't be in the cache when they are requested), and hence the queue is updated accordingly. Also, for $\frac{k}{2} \leq j < k$ the first player needs to access page j at time $k + 2j$. Since till this time there has been only $j - \frac{k}{2}$ page faults of the second player in this phase, and according to the order of the pages in the queue at the beginning of the phase, it follows that r_{k+2j} will not incur a page fault. Accordingly, after r_{3k-1} is served, the pages $k+1, k+2, \dots, \frac{3k}{2}, k, 1, 2, \dots, \frac{k}{2} - 1$ are in the queue in this order. Thus, r_{3k} does not incur a page fault, but r_{3k+1} does.

Now, we have the following proposition which completes the proof.

Proposition 6. *After r_{3k+1} is served and till the end of the sequence, the cache contains the k most recent requests that were designated to the second player. Specifically, these requests are for different pages and no page faults are incurred by the requests designated to the the first player.*

Proof. By induction. After r_{3k+1} is served the claim holds by the above discussion.

Notice that every request r_i, $i > 3k$, that is designated to the first player is for $r_{i-(2k-1)}$, which is one of the k most recent requests that were designated to the second player. Hence, by the induction hypothesis, it is in the cache and incurs no page fault.

As for the requests r_i that are designated to the second player, $i > 3k+1$, it holds that $r_i = r_{i-(k+1)}$. By Lemma 4, it holds that $r_{i-(k+1)} \notin \{r_{i-j}\}_{j=2}^{k}$ and $r_{i-(k+1)} \notin \{r_{i-(k+j)}\}_{j=2}^{k}$. As the k most recent requests that were designated to the second player are r_{i-j} for even $2 \leq j \leq 2k$, the claim follows. $\square$

$\square$

$\square$

4 The Static List Accessing Game

In this section we study another simple online game which generalizes the well known static list accessing online problem. In this online game a list is shared between N players (for example, it is located on a server and N different players access it). Given a request r_i, player $p(r_i)$ should access the given element in the list. Each player is paying for his actions (accessing and paid transpositions).

4.1 Lower Bounds

In general, the known lower bounds for the online problem are all valid for the online game, including the lower bound of $2 - \frac{2}{\ell+1}$ on the competitiveness of any deterministic online algorithm (see [7]), and the lower bound of $\frac{3}{2} - \frac{5}{\ell+5}$ on any randomized algorithm against oblivious adversaries ([18]).

4.2 Algorithm Move-to-Front

One of the well-known online algorithms for the static list accessing problem is Move-to-Front ($\mathcal{MTF}$), which is also a a valid strategy in the online game. When using this algorithm, after every access to an item, the item is brought to the front of the list.

Theorem 7. *$\mathcal{MTF}$ is $(2 - \frac{1}{\ell})$-self-competitive where ℓ is the length of the given list.*

Proof. The proof follows the same ideas as the proof of the competitiveness of $\mathcal{MTF}$ (see [7]). We sketch the proof. At the heart of the proof of $\mathcal{MTF}$ is a potential function, Φ, which is defined to be the number of inversions between the list of the player playing $\mathcal{MTF}$ and the list of the player playing $\mathcal{OPT}$.

Fix a player k, and assume that all other players use $\mathcal{MTF}$ as their strategy. Similarly to the proof of the competitiveness of $\mathcal{MTF}$, for each request designated to the kth player, it holds that the cost paid by $\mathcal{MTF}$ plus the change in the potential function is bounded by, roughly speaking, twice the cost paid by $\mathcal{OPT}$. For each request designated to a player different from k, the cost for the kth player is 0, regardless of his strategy, and the value of the potential function can only decrease. Hence, by the potential function argument, the total cost paid by $\mathcal{MTF}$ is at most $(2 - \frac{1}{\ell})$ times the cost paid by $\mathcal{OPT}$. $\square$

The lower bounds in the general case imply that $(\mathcal{MTF}, \ldots, \mathcal{MTF})$ is a $(1 + \Theta(\frac{1}{\ell}))$-equilibrium with respect to deterministic strategies, and a $(\frac{4}{3} + \Theta(\frac{1}{\ell}))$-equilibrium against an oblivious adversary.

4.3 Algorithm TIMESTAMP

We now turn to examine another well known deterministic online algorithm, Algorithm TIMESTAMP, which is known to be 2-competitive (see [7]). The algorithm is as follows.

ALGORITHM TIMESTAMP: Upon receiving a request for item x, insert x in front of the first (from the front of the list) item y that precedes x on the list and was requested at most once since the last request for x. Do nothing if there is no such item y, or if x is requested for the first time.

In contrast with $\mathcal{MTF}$, it is not obvious how to use algorithm TIMESTAMP to derive an online strategy for the list accessing game. In what follows we discuss one possibility, in which each player applies the algorithm with respect to his requests only, while ignoring requests served by other players. We denote the resulting strategy by $\mathcal{TS}$. This approach is vital when the players do not know the other players' requests (notice that this is not required to apply the $\mathcal{MTF}$ strategy).

Theorem 8. *For every i and for $s_{-i} = (\mathcal{TS}, \ldots, \mathcal{TS})$, $\mathcal{R}_{s_{-i}}(\mathcal{TS})$ is unbounded.*

Proof. Assume that we are given a list which is initially ordered as follows: $x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$. Consider the following request sequence and the cost for the first player, assuming that the second player uses the $\mathcal{TS}$ strategy.

Initialization phase:

		$\mathcal{TS}$		$\mathcal{MTF}$	
r_i	$p(r_i)$	Cost	New configuration	Cost	New configuration
x_1	1	1	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$	1	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$
y_1	2	–	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$	–	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$
x_1	1	1	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$	1	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$
y_2	2	–	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$	–	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_1	1	1	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$	1	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$
y_k	2	–	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$	–	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$
x_1	1	1	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$	1	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$
x_k	2	–	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$	–	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$
x_1	1	1	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$	1	$x_1, x_2, \ldots, x_k, z, y_1, \ldots, y_k$
x_k	2	–	$x_k, x_1, x_2, \ldots, x_{k-1}, z, y_1, \ldots, y_k$	–	$x_k, x_1, x_2, \ldots, x_{k-1}, z, y_1, \ldots, y_k$

The main phase:

x_k	1	1	$x_k, x_1, x_2, \ldots, x_{k-1}, z, y_1, \ldots, y_k$	1	$x_k, x_1, x_2, \ldots, x_{k-1}, z, y_1, \ldots, y_k$
x_{k-1}	2	–	$x_k, x_1, x_2, \ldots, x_{k-1}, z, y_1, \ldots, y_k$	–	$x_k, x_1, x_2, \ldots, x_{k-1}, z, y_1, \ldots, y_k$
x_k	1	1	$x_k, x_1, x_2, \ldots, x_{k-1}, z, y_1, \ldots, y_k$	1	$x_k, x_1, x_2, \ldots, x_{k-1}, z, y_1, \ldots, y_k$
x_{k-1}	2	–	$x_{k-1}, x_k, x_1, \ldots, x_{k-2}, z, y_1, \ldots, y_k$	–	$x_{k-1}, x_k, x_1, \ldots, x_{k-2}, z, y_1, \ldots, y_k$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_2	1	1	$x_2, \ldots, x_k, x_1, z, y_1, \ldots, y_k$	1	$x_2, \ldots, x_k, x_1, z, y_1, \ldots, y_k$
x_1	2	–	$x_2, \ldots, x_k, x_1, z, y_1, \ldots, y_k$	–	$x_2, \ldots, x_k, x_1, z, y_1, \ldots, y_k$
x_2	1	1	$x_2, \ldots, x_k, x_1, z, y_1, \ldots, y_k$	1	$x_2, \ldots, x_k, x_1, z, y_1, \ldots, y_k$
x_1	2	–	$x_1, \ldots, x_k, z, y_1, \ldots, y_k$	–	$x_1, \ldots, x_k, z, y_1, \ldots, y_k$

z	1	$k+1$	$x_1,\dots,x_k,z,y_1,\dots,y_k$	$k+1$	$z,x_1,\dots,x_k,y_1,\dots,y_k$
y_1	2	$-$	$x_1,\dots,x_k,y_1,z,y_2,\dots,y_k$	$-$	$y_1,z,x_1,\dots,x_k,y_2,\dots,y_k$
y_1	1	$k+1$	$x_1,\dots,x_k,y_1,z,y_2,\dots,y_k$	1	$y_1,z,x_1,\dots,x_k,y_2,\dots,y_k$
y_2	2	$-$	$x_1,\dots,x_k,y_2,y_1,z,y_3,\dots,y_k$	$-$	$y_2,y_1,z,x_1,\dots,x_k,y_3,\dots,y_k$
y_2	1	$k+1$	$x_1,\dots,x_k,y_2,y_1,z,y_3,\dots,y_k$	1	$y_2,y_1,z,x_1,\dots,x_k,y_3,\dots,y_k$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
y_k	2	$-$	$x_1,\dots,x_k,y_k,\dots,y_1,z$	$-$	$y_k,\dots,y_1,z,x_1,\dots,x_k$
y_k	1	$k+1$	$x_1,\dots,x_k,y_k,\dots,y_1,z$	1	$y_k,\dots,y_1,z,x_1,\dots,x_k$
x_k	2	$-$	$x_1,\dots,x_k,y_k,\dots,y_1,z$	$-$	$x_k,y_k,\dots,y_1,z,x_1,\dots,x_{k-1}$
x_k	1	k	$x_k,x_1,\dots,x_{k-1},y_k,\dots,y_1,z$	1	$x_k,y_k,\dots,y_1,z,x_1,\dots,x_{k-1}$
x_k	2	$-$	$x_k,x_1,\dots,x_{k-1},y_k,\dots,y_1,z$	$-$	$x_k,y_k,\dots,y_1,z,x_1,\dots,x_{k-1}$

The total cost for the first player when he uses $\mathcal{TS}$ is $k+2k+(k+1)+k(k+1)+k$, while the total cost when he uses $\mathcal{MTF}$ is $k+2k+(k+1)+k+1$. We get a ratio of $\Omega(k)=\Omega(\ell)$. The same ideas can be used to generate an arbitrary long sequence by repeating a variant of the main phase. After the first part of the main phase we get:

x_2	1	1	$x_2,\dots,x_k,x_1,y_k,\dots,y_1,z$	1	$x_2,\dots,x_k,y_k,\dots,y_1,z,x_1$
x_1	2	$-$	$x_2,\dots,x_k,x_1,y_k,\dots,y_1,z$	$-$	$x_2,\dots,x_k,x_1,y_k,\dots,y_1,z$
x_2	1	1	$x_2,\dots,x_k,x_1,y_k,\dots,y_1,z$	1	$x_2,\dots,x_k,x_1,y_k,\dots,y_1,z$
x_1	2	$-$	$x_1,\dots,x_k,y_k,\dots,y_1,z$	$-$	$x_1,\dots,x_k,y_k,\dots,y_1,z$

Hence, we need the following change so that the $\mathcal{TS}$ player will not insert z before x_1:

x_1	1	1	$x_1,\dots,x_k,y_k,\dots,y_1,z$	1	$x_1,\dots,x_k,y_k,\dots,y_1,z$
x_1	2	$-$	$x_1,\dots,x_k,y_k,\dots,y_1,z$	$-$	$x_1,\dots,x_k,y_k,\dots,y_1,z$
x_1	1	1	$x_1,\dots,x_k,y_k,\dots,y_1,z$	1	$x_1,\dots,x_k,y_k,\dots,y_1,z$
x_1	2	$-$	$x_1,\dots,x_k,y_k,\dots,y_1,z$	$-$	$x_1,\dots,x_k,y_k,\dots,y_1,z$
z	1	$2k+1$	$x_1,\dots,x_k,z,y_k,\dots,y_1$	$2k+1$	$z,x_1,\dots,x_k,y_k,\dots,y_1$
y_1	2	$-$	$x_1,\dots,x_k,y_1,z,y_k,\dots,y_2$	$-$	$y_1,z,x_1,\dots,x_k,y_k,\dots,y_2$
y_1	1	$k+1$	$x_1,\dots,x_k,y_1,z,y_k,\dots,y_2$	1	$y_1,z,x_1,\dots,x_k,y_k,\dots,y_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
y_k	2	$-$	$x_1,\dots,x_k,y_k,\dots,y_1,z$	$-$	$y_k,\dots,y_1,z,x_1,\dots,x_k$
y_k	1	$k+1$	$x_1,\dots,x_k,y_k,\dots,y_1,z$	1	$y_k,\dots,y_1,z,x_1,\dots,x_k$
x_k	2	$-$	$x_1,\dots,x_k,y_k,\dots,y_1,z$	$-$	$x_k,y_k,\dots,y_1,z,x_1,\dots,x_{k-1}$
x_k	1	k	$x_k,x_1,\dots,x_{k-1},y_k,\dots,y_1,z$	1	$x_k,y_k,\dots,y_1,z,x_1,\dots,x_{k-1}$
x_k	2	$-$	$x_k,x_1,\dots,x_{k-1},y_k,\dots,y_1,z$	$-$	$x_k,y_k,\dots,y_1,z,x_1,\dots,x_{k-1}$

$\square$

Intuitively, the problem with the $\mathcal{TS}$ startegy, in contrast to $\mathcal{MTF}$, is that its decisions depend on the current configuration of the list. This configuration can be changed by a player so as to manipulate other players to play in accordance with his interests.

4.4 Algorithm BIT

Algorithm $\mathcal{BIT}$ is a randomized online algorithm for list accessing problem. For each element x in the list, a bit $b(x)$ is randomly initialized. When a request to access element x arrives, bit $b(x)$ is complemented, and if $b(x) = 1$ then the element is moved to the front of the list. If every player maintains a separate set of random bits, and uses those bits to implement a strategy following the ideas of Algorithm $\mathcal{BIT}$, the resulting strategy is self-competitive. The analysis of Algorithm $\mathcal{BIT}$ can be easily extended to prove that the $\mathcal{BIT}$ strategy is $\frac{7}{4}$-self competitive. Thus, we conclude that $(\mathcal{BIT}, \ldots, \mathcal{BIT})$ is a $(\frac{7}{6} + \Theta(\frac{1}{\ell}))$-Nash Equilibrium against an oblivious adversary.

5 Metrical Task Games

We consider the metrical task game in this section as an abstract model for analyzing online games. The metrical task game, generalizing metrical task systems, captures a wide range of cost minimization games.

5.1 The Model

In what follows we use terminology similar to that of [7, Chapter 9]. A *metrical task game* consists of a set of N players, a set of points (configurations/states) S, and for each player i, a distance function $d_i : S \times S \to \mathbb{R}^+ \cup \{\infty\}$ that satisfies triangle inequality, and a set of allowable tasks $\mathcal{R}_i \in \mathbb{R}^{|S|}$. We note that for the sake of generalization we do not require symmetry, reflexivity, or positivity of the distance function (although this means that the name is a bit misleading). A legal request in a metrical task game is a task r such that $r \in \mathcal{R}_{p(r)}$. Roughly speaking, a strategy of player i is a function that assigns a target configuration for every *history*, which is a sequence $\{(q_{j-1}, r_j)\}_{j>0}$ of pairs of a configuration and a legal request.

Given an initial configuration and a sequence of legal requests, each request is served by the player it is designated to. While a request is being served, the configuration of the system can be changed by other players. If at the time that request r is invoked, the current configuration of the system is q, then player $p(r)$ can first change the configuration of the system to any desired configuration q' such that $d_i(q, q') < \infty$ (and sometimes we also require that $r(q') < \infty$), incurring a transition cost of $d_i(q, q')$. Then, the player serves the task in state q', incurring a processing cost of $r(q')$.

5.2 The Non-malleability Principle

While looking for (approximate) equilibria in a metrical task game, we face two related difficulties. The first difficulty comes from the search for a competitive algorithm for each player. The second difficulty is guaranteeing that an equilibrium results from the interaction between the algorithms of the players . While solving the first difficulty is somewhat similar to designing competitive algorithms, the second one seems to be inherently different from any other familiar algorithmic problem[3]. Thus, coping with both difficulties at once might be a hard task. A possible way to get around this hardness is by developing techniques that will enable us to cope with each part separately. Motivated by this idea, we examine the interaction between online strategies, and in particular the impact of this interaction on the performance of each strategy.

The decisions made by online algorithms can depend on the current configuration of the system. When online strategies interact, they might be sensitive to configuration changes caused by other players. Moreover, a player using an online algorithm might be manipulated by other players who change the configuration

[3]The paging game seems to be an exception in which a known competitive algorithm leads to a simple solution for the problem.

of the system wisely, and in particular by an adversary who can construct the request sequence in advance. Consequently, an algorithm which is competitive with respect to the online problem induced by S, d_i and $\mathcal{R}_i$, might yield a strategy which is not (self-) competitive due to the interaction between the players, and the fact that other players can be manipulated by the adversary. The $\mathcal{FIFO}$ strategy for the paging game is an example in which the adversary can construct a request sequence in which, by evicting pages not according to the $\mathcal{FIFO}$ policy, it causes other players to evict pages different from those it needs to serve, and hence avoids page faults. From the perspective of the players, the outcome is the poor performance of $\mathcal{FIFO}$ that is reflected in Theorem 3. In order to overcome the above phenomenon, and with the goal of deriving online strategies that are in equilibrium from competitive online algorithms in mind, we suggest the *non-malleability principle*. The idea behind the principle is to prevent players from manipulating the game.

THE NON-MALLEABILITY PRINCIPLE. *A strategy vector s complies with the non-malleability principle if for every i, strategy s_i can be described as a function of the request sequence and the initial configuration only, and it is independent of the strategies of the other players.*

The non-malleability principle can be said to nullify interaction between players, and thus the problem of finding equilibria can be reduced to the problem of finding for each player a “good” strategy. Specifically, each player actually faces an induced online problem - the metrical task system MTS_i induced by S, d_i and $\mathcal{R}_i$ - but with the exception that there might be some external configuration changes in between requests. These changes do not incur any cost to the player and they are independent of the algorithm the player uses. Designing competitive algorithms for this kind of online problem can be done either from scratch or based on known competitive algorithms for metrical task system without the external configuration changes.

5.3 The Optimistic Method

We present here the *optimistic method* which implements the non-malleability principle, together with two techniques that are used in different cases and are based on this method.

The optimistic method constructs a strategy vector s in which all players make their *decisions* while assuming that the current configuration is some *anticipated configuration*, and ignoring the actual current configuration. Obviously, such a strategy vector complies with the non-malleability principle. Due to performance considerations, we would like the anticipated configuration to be the current configuration, and thus we set the anticipated configuration to be the configuration resulting from the players playing according to s. In order to apply the optimistic method it is usually required that every request is publicly known once it is invoked, as every player must know all past requests of all the players in order to calculate the anticipated configuration and implement its strategy.

The difficulty in implementing the optimistic method originates from the need to make sure that the resulting strategies are feasible, since ignoring the current configuration might lead to illegal configuration changes by the players. We devise two techniques that are based on the optimistic method, where each technique can be applied to a different kind of online games. Both techniques reduce the problem of constructing an equilibrium to the problem of designing competitive algorithms for the induced online problems $\{MTS_i\}_i$. These algorithms are then converted to strategies by operating according to the anticipated configuration instead of according to the current one. In the *adjustment technique* the anticipated configuration is calculated in a straightforward manner, and hence, due to feasibility considerations, the applicability of the technique is rather limited. On the other hand, the structure of the games on which the *indifference technique* is applied facilitates the use of a more relaxed approach that still guarantees feasibility.

5.3.1 The Adjustment Technique

In the adjustment technique, the players determine the configuration changes independently of the current configuration. As a result, after each player serves a request, the configuration depends only on the request sequence (and maybe the initial configuration), and thus it can be said to be adjusted to s. This adjustment is crucial when one of the players deviates from s, as this deviation would not have further effect on the system once an adjustment move is made.

Formally, given a strategy vector s, a request sequence $\sigma = \{r_j\}_j$ and an initial configuration q_0, we recursively define the history generated by s on the request sequence σ when initiated at q_0 as $h(\sigma, s, q_0) = \{(q_{j-1}, r_j)\}_j$, where for every $j > 0$, $q_j = s_{p(r_j)}(h(pref_j(\sigma), s, q_0))$ is the configuration of the system after player $p(r_j)$ serves the jth request given that all the players played according to s. Now, let s_i^{adj} be a strategy for the ith player defined with respect to s as follows: $s_i^{adj}(h(\sigma, s', q_0)) \triangleq s_i(h(\sigma, s, q_0))$ for every σ, s' and q_0.

THE ADJUSTMENT TECHNIQUE. *Given a strategy vector s, construct a strategy vector s^{adj} by replacing each strategy s_i with the strategy s_i^{adj}.*

To ensure feasibility of s^{adj}, the state transitions made by s_i^{adj}, for every i, must be feasible regardless of the strategies played by the other players. Hence, the applicability of the adjustment technique depends on the metrical task game and the strategy vector s. A metrical task game on which the adjustment technique can be applied to every strategy vector s is a game in which all transition costs are finite, i.e., $d_i(q, q') < \infty$ for every player i, and configurations q and q'. We call such a metrical task game a *total sharing game*.

The adjustment technique yields equilibrium if it is applied to competitive algorithms for the induced online problems, $\{MTS_i\}_i$, with the possible external configuration changes. Such algorithms can be based on competitive algorithms for $\{MTS_i\}_i$ (without external configuration changes), if they are adapted to external configuration changes. A simple way to adapt an online algorithm $\mathcal{A}$ to external configuration changes is by resetting the online algorithm after every external configuration change, i.e., following a request allocated to a different player in the game, the player forgets about any past requests and serves the next request as if it was the first request allocated to it. Formally, abusing notation, we denote by $\mathcal{A}^{adj}$ the strategy for the metrical task game obtained when applying both this modification and the adjustment technique.

Theorem 9. *Let $\mathcal{G}$ be a total sharing metrical task game, and let $\{\mathcal{A}_i\}_{1\leq i\leq N}$ be online algorithms such that for every i, $\mathcal{A}_i$ is strictly α_i-competitive online algorithm for MTS_i for any initial configuration $q_0 \in S$. Then, for every i, $\mathcal{A}_i^{adj}$ is strictly α_i-competitive with respect to the strategy vector $\{\mathcal{A}_j^{adj}\}_{-i}$ in $\mathcal{G}$.*

Proof. Fix i and assume that all the players except the ith player are playing according to the strategy vector $\{\mathcal{ALG}_j^{adj}\}_{-i}$. Fix a request sequence σ, and let $\mathcal{OPT}(\sigma)$ denote an optimal strategy for the ith player for serving σ. Denote by $\mathcal{ALG}(\sigma)$ the cost incurred by the strategy $\mathcal{ALG}$ while serving σ in $\mathcal{G}$. Let $\sigma_j = r_{j_s}, r_{j_s+1}, \ldots, r_{j_s+|\sigma_j|-1}$ be the jth maximal subsequence of requests in σ allocated to the ith player (i.e. $p(r_{j_s-1}) \neq i$ and $p(r_{j_s+|\sigma_j|}) \neq i$ for every j). By the definition of j_s and the adjustment technique, q_{j_s-1} - the configuration of the system when the j_sth request is invoked - is fully defined by σ and the strategy vector $\{\mathcal{ALG}_j^{adj}\}_{-i}$, regardless of the ith player strategy.

Denote by $\mathcal{ALG}(\sigma, q)$ the cost incurred by the online algorithm $\mathcal{ALG}$ while serving σ in MTS_i when the initial state is q. Finally, denote by $\mathcal{OPT}(\sigma, q)$ the optimal solution for serving σ in MTS_i when the initial state is q.

Then, $\mathcal{A}_i^{adj}(\sigma) = \sum_j \mathcal{A}_i(\sigma_j, q_{j_s-1}) \leq \sum_j \alpha_i \cdot \mathcal{OPT}(\sigma_j, q_{j_s-1}) \leq \alpha_i \cdot \mathcal{OPT}(\sigma)$, where the equality follows from the definition of the strategy $\mathcal{A}_i^{adj}$, the first inequality follows from the competitiveness of $\mathcal{A}_i$ and the last inequality follows as the optimal strategy induces solutions for the online problems $\{(\sigma_j, q_{j_s-1})\}_j$. $\square$

By applying Theorem 9 on $\mathcal{FIFO}$ we get a self-competitive strategy as the paging game is a total sharing metrical task game and $\mathcal{FIFO}$ is a strictly competitive for the paging problem. Notice that Theorem 9 can be generalized for every metrical task game and strategy vector to which the adjustment technique can be applied. Also note that the adjustment technique results in strategies that maintain their competitiveness given any history, and hence it can be used to construct sub-game perfect (approximate)-equilibria.

5.3.2 The Indifference Technique

We define a *product metrical task game* to be a game satisfying the following two properties. First, its configuration set can be written as $S = \times_{1 \le i \le N} S_i$, and hence every configuration q can be described by a vector of length N, $q = (q_1, \ldots, q_N)$, where $q_i \in S_i$. Second, every distance function d_i satisfies for every q and q', such that there exists $j \ne i$ with $q_j \ne q'_j$, $d_i(q, q') = \infty$.

The second property implies that a player can only change "its" coordinate in the configuration vector, and thus any strategy of the ith player can be described as a function that assigns $q_i \in S_i$ for every history. Thus, using similar notations, $h(\sigma, s, q_0) = \{(q_{j-1}, r_j)\}_j$, where for every $j > 0$, $(q_j)_{p(r_j)} = s_{p(r_j)}(h(pref_j(\sigma), s, q_0))$ and $(q_j)_k = (q_{j-1})_k$ for every $k \ne p(r_j)$. Now, let s_i^{ind} be a strategy for the ith player defined with respect to s as follows: $\forall \sigma, s', q_0,\ s_i^{ind}(h(\sigma, s', q_0)) \triangleq s_i(h(\sigma, s, q_0))$.

THE INDIFFERENCE TECHNIQUE. *Given a strategy vector s, construct a strategy vector s^{ind} by replacing each strategy s_i with the strategy s_i^{ind}.*

As the indifference technique complies with the non-malleability principle, once it is applied, each player faces an induced online problem. The special structure of product metrical task games yields an interesting structure of the induced problems - each can be viewed as a metrical task system with configuration set S_i in which the distance function and allowable tasks can vary over time (but there are no external configuration changes). Formally, for player i, each combination of the values of the $N-1$ coordinates in a configuration corresponding to the players different from i induces a distance function and a set of allowable tasks for player i. As the other players change some coordinates of the configuration, the distance function and the allowable tasks for the ith player are changed. This description of induced metrical task systems might appear a bit strange, but it turns out that certain competitive algorithms for known metrical task systems can be adapted to these distance function and allowable tasks changes. In such cases, the indifference technique yields (approximate) equilibria.

6 The Caching Game

In the caching game there are N players, each representing a server in a network given by an undirected edge-weighted graph $G = (V, E)$ ($|V| = N$). Given is a set F of m different files that are assumed to have unit size. Server i has a cache of capacity k_i which is initially empty. When a server gets a request for a file it should serve the request by accessing the file. The server may cache the file in its own cache before serving the request. We assume that caching a file incurs some constant cost D, while accessing a remote replica costs the accessing server exactly the distance between the server and the accessed replica. If a requested file is not cached by any server in the network, it must be cached by the server that serves the request.

Lemma 10. *For every i such that $k_i = \infty$, strategy vector s_{-i}, and deterministic strategy $\mathcal{A}$, $\mathcal{R}_{s_{-i}}(\mathcal{A}) \ge 2$.*

Proof. Consider a sequence of repeated requests to the same file by player i. Assume that initially the file is not cached in server i, but is cached in server j whose distance from server i is exactly 1. Then, for the ith server the problem is reduced to the ski-rental problem with buying price of D, and the theorem follows from the lower bound on the competitiveness of deterministic online algorithms for the ski-rental problem. □

Lemma 11. *For every i such that $k_i < \infty$, strategy vector s_{-i}, and deterministic strategy $\mathcal{A}$, $\mathcal{R}_{s_{-i}}(\mathcal{A}) \geq k_i$.*

Proof. Consider a sequence of requests, all designated to server i, and all for files that are initially not cached by any server. Then, for the ith server, the problem is reduced to the paging problem with cache size of k_i and the theorem follows from the lower bound on the competitiveness of deterministic online algorithms for the paging problem. □

We now introduce a deterministic strategy which is $O(k_i)$-competitive for $k_i < \infty$ with respect to the constructed strategy vector and 2-competitive for $k_i = \infty$ with respect to any strategy vector. It is not hard to show that the caching game is a product game, allowing us to use the indifference technique. It suffices to provide a competitive algorithm for the induced online problem of every player, as follows. A server gets a sequence of requests for files. At each step, the requested file can be first cached by the server incurring a cost of D, subject to the cache size constraint. Then, the request is served by either accessing a locally cached copy (if such copy exists), incurring no cost, or accessing a remote replica, incurring a cost that is given as part of the request. Note that if the file is not cached by any server, the remote accessing cost can be set to ∞. Note that the same file may be requested with different remote accessing costs.

Observe that the above induced problem generalizes both the online paging problem and the ski-rental problem. Thus, we combine the marking principle borrowed from the online paging algorithms with the rent-or-buy principle to obtain the following algorithm, which we refer to as algorithm $\mathcal{COUNT}$.

In algorithm $\mathcal{COUNT}$, the server maintains a counter for each file in F. The counter is initialized to 0. We denote the value of the counter of file f by $v(f)$. When a request for file f arrives, the server increases $v(f)$ by the remote accessing cost indicated along with the request. If the requested file is in the cache, the server accesses it directly, incurring no cost. Otherwise, if $v(f) < D$, the file is accessed remotely, while if $v(f) \geq D$, f is to be cached by the server. If the cache is full, then the page with the lowest counter value among the cached files is to be evicted. If the value of the counter of the evicted page is $\geq D$, then the counters of all the files in the cache (including both f and the evicted file) are reset to 0.

Theorem 12. *In the induced online problem, algorithm $\mathcal{COUNT}$ has a competitive ratio of at most $4k_i + 6$ for $k_i < \infty$, and at most 2 for $k_i = \infty$.*

Proof. For $k_i = \infty$, there is no eviction of pages, and the problem reduces to the ski-rental problem. The theorem follows as algorithm $\mathcal{COUNT}$ reduces to the 2-competitive algorithm for the ski-rental problem.

Now assume that $k_i < \infty$. Fix a request sequence σ, and divide it into phases. The first phase begins at the first request. A phase ends right after a request that caused algorithm $\mathcal{COUNT}$ to reset the counters of the files in the cache. Let ℓ be the number of phases and denote by F_j the set of files such that the value of their counters has exceeded D during phase j (notice that the counter of these files are reset before the beginning of the $(j+1)$st phase). Let $v_{final}(f)$ be the value of the counter of file f after $\mathcal{COUNT}$ finished to serve σ. Then, by the definition of $\mathcal{COUNT}$, $\mathcal{COUNT}(\sigma) \leq \sum_{j=1}^{\ell} 2D|F_j| + \sum_{f \notin F_\ell} v_{final}(f) = 2D(\ell - 1)(k_i+1) + 2D|F_\ell| + \sum_{f \notin F_\ell} v_{final}(f)$. Since $\mathcal{OPT}$ must pay at least $v_{final}(f)$ for serving the requests for every file $f \notin F_\ell$, and at least D for every file $f \in F_\ell$, we get that $2D|F_\ell| + \sum_{f \notin F_\ell} v_{final}(f) \leq 2\mathcal{OPT}(\sigma)$.

It is left to show that $D(\ell - 1) \leq 2\mathcal{OPT}(\sigma)$. Notice that for $j < \ell$, there is a file $f_j \in F_j$ that is not in $\mathcal{OPT}$'s cache at the beginning of phase j, as there are $k_i + 1$ files in F_j. We refer to f_j as the *charging file* of phase j. For every $f \in F$, let $h(f)$ be the number of phases such that f is their charging file. Notice that $\sum_{f \in F} h(f) \geq (\ell - 1)$. Obviously, for a file f with $h(f) \geq 1$, $\mathcal{OPT}$ pays at least D while serving the requests for file f till the end of the first phase of which f is a charging file. For a file f with $h(f) \geq 2t + 1$ for some $t > 0$, $\mathcal{OPT}$ pays at least D while serving the requests for file f between the beginning of the $2t$th phase of which f is a charging file till the end of the $(2t+1)$th phase of which f is a charging file, as f is not in $\mathcal{OPT}$'s cache at the beginning of the $2t$th phase of which f is a charging file, and the total remote accessing cost for these requests is at least D. It follows that $\mathcal{OPT}(\sigma) \geq \sum_{f \in F} D \cdot \frac{h(f)}{2} = \frac{D}{2} \cdot \sum_{f \in F} h(f) \geq \frac{D}{2} \cdot (\ell - 1)$. $\square$

If no server has a cache capacity constraint, then by a careful analysis we get that $(\mathcal{COUNT}, \ldots, \mathcal{COUNT})$ is an equilibrium with respect to to deterministic strategies. To achieve competitive strategies in the caching game for the case that there exists some i for which $k_i < \infty$, we further apply the indifference technique. Then, letting $s = (\mathcal{COUNT}^{ind}, \ldots, \mathcal{COUNT}^{ind})$, we have that $\mathcal{R}_{s_{-i}}(\mathcal{COUNT}^{ind}) \leq 4k_i + 6$ for $k_i < \infty$ and $\mathcal{R}_{s_{-i}}(\mathcal{COUNT}^{ind}) \leq 2$ for $k_i = \infty$. Hence, s is a 10-equilibrium with respect to deterministic strategies.

7 The Online Generalized Steiner Game

In the *Online Generalized Steiner Game* N players are given an undirected graph $G = (V, E)$ with an associated edge cost function $c : E \to \mathbb{R}^+$. A request sequence σ arrives online. Each request in σ is of the form (a, b), where $a, b \in V$, and it is allocated to one of the N players. During the game, the players construct F, a subgraph of G, whose edge set is referred to as the *solution* of the game. Initially, the solution is empty. Upon arrival of request $r_i = (a, b)$, player $p(r_i)$ serves the request by adding edges to the solution so that vertices a and b are connected in F. When a player adds edges to the solution, it pays their cost. Accordingly, the goal of each of the players is to minimize the cost it pays during the game.

As a first step towards constructing an equilibrium of this game, consider some known algorithms for the online problem. A greedy algorithm for the on-line generalized Steiner tree problem is $O(\log^2 n)$-competitive [3]. In [5], Berman and Coulston presented an $O(\log n)$ competitive on-line algorithm (denote it by $\mathcal{BC}$), matching the $\Omega(\log n)$ lower bound of Imaze and Waxman [13].

The following results point out the inefficiency in using those algorithms to construct an equilibrium. Specifically, we construct examples in which the adversary can "manipulate" the other players, resulting poor performance by the players.

Lemma 13. *The greedy algorithm is not self-competitive.*

Proof. Consider the following graph, $G = (V, E)$:

$$V = \{v_0, v_1, v_2\} \cup \{u_i | 1 \leq i \leq k\} \cup \{w_i | 1 \leq i \leq k\}$$

$$E = \{(v_0, v_1), (v_1, v_2)\} \cup \{(v_1, u_i) | 1 \leq i \leq k\} \cup \{(u_i, w_i) | 1 \leq i \leq k\} \cup \{(w_i, v_2) | 1 \leq i \leq k\},$$

where the weight of the edges (u_i, w_i) is 5 and the weight of all the other edges are 2. Now, consider the following request sequence and the cost for the first player:

		Greedy		*OPT*	
r_i	$p(r_i)$	edges added	$Greedy$'s cost	edges added	OPT's cost
v_0, v_1	1	(v_0, v_1)	2	$(v_0, v_1), (v_1, v_2)$	4
u_1, w_1	2	(u_1, w_1)	–	$(v_1, u_1), (w_1, v_2)$	–
v_1, u_1	1	(v_1, u_1)	2	–	0
u_2, w_2	2	(u_2, w_2)	–	$(v_1, u_2), (w_2, v_2)$	–
v_1, u_2	1	(v_1, u_2)	2	–	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
u_k, w_k	2	(u_k, w_k)	–	$(v_1, u_k), (w_k, v_2)$	–
v_1, u_k	1	(v_1, u_k)	2	–	0

The total cost for a greedy player is hence $2(k+1)$, while the total cost for the optimal player is 4, resulting a competitive ratio of at least $\Omega(n)$. □

Lemma 14. *Algorithm $\mathcal{BC}$ is not self-competitive.*

Proof. The straight forward implementation of $\mathcal{BC}$ as a strategy implies that a player may buy many edges besides the edges that are needed to connect a given pair. Hence an example in which such implementation is inefficient can be easily constructed:

$$V = \{v_0\} \cup \{u_i | 1 \leq i \leq k\} \cup \{w_i | 1 \leq i \leq k\}$$

$$E = \{(v_1, u_i) | 1 \leq i \leq k\} \cup \{(u_i, w_i) | 1 \leq i \leq k\},$$

where the weight of the all the edges is 1 and the request sequence is:

r_i	$p(r_i)$	Edges added
u_1, w_1	2	(u_1, w_1)
u_2, w_2	2	(u_2, w_2)
$\vdots$	$\vdots$	$\vdots$
u_{k-1}, w_{k-1}	2	(u_{k-1}, w_{k-1})
v_1, u_k	1	(v_1, u_k) and $\mathcal{BC}$ also adds the edges (v_1, u_i) for $1 \leq i < k$

The cost for a $\mathcal{BC}$ player is hence k, while the cost for the optimal player is 1, resulting a competitive ratio of at least $\Omega(n)$.

Another implementation of $\mathcal{BC}$ might maintain a seperate set of connected components for each player. Such a strategy is inefficient as can be seen by a simple modification to the example given in the previous lemma, in which instead of assigning all the (u_i, w_i) requests to a single player (player 2), each request is designated to a different (new) player. The rest of the analysis is the same by the definition of $\mathcal{BC}$. □

Despite the above examples, in what follows we show that $\mathcal{BC}'$, a slight modified version of $\mathcal{BC}$, is a competitive algorithm for the players' induced problems in the game. It is not hard to show that this game is a product game, and hence the indifference technique can be applied to $\mathcal{BC}'$ to obtain an $O(\log n)$-self-competitive strategy. We note that we do not know whether a similar result can be obtained using the greedy approach.

7.1 Algorithm $\mathcal{BC}'$

Notice that the induced problem is identical to the online generalized Steiner problem with the exception that before each request arrives, some edge costs can get nullified. In what follows, we assume that each edge having zero cost is added to the solution immediately. We denote by G^i the graph G with the edge costs upon arrival of the ith request.

In what follows we describe the differences between $\mathcal{BC}'$ and $\mathcal{BC}$ as it appears in [5][Sect. 2]. The distance between two vertices in the graph is calculated as if the edges that were chosen to the solution have zero cost. We will denote the distance between vertices a and b just before serving the ith request by $d^i(a,b)$. If algorithm $\mathcal{BC}'$ decides in step 3.4 (respectively, step 4.4) to add a path between u and v to the solution, then it adds to the solution all the edges that have not yet been added to the solution among those of some shortest path between u and v. Algorithm $\mathcal{BC}'$ maintains a set of connected components denoted by C, which is initially empty. During steps 3 and 4 of the algorithm, $\mathcal{BC}'$ examines only the connected components in C. Given a request $r_i = (a,b)$ it adds two new singletons, $\{a\},\{b\}$, to C. If, in step 3.4 (respectively, step 4.4), $\mathcal{BC}'$ adds to the solution a path P, then it updates C by merging the connected components of all the vertices in P into one new connected component that also includes all the vertices of the path P that did not belong to some connected component in C.

7.2 Analysis of Algorithm $\mathcal{BC}'$

Fix a request sequence σ. We denote by OPT the set of edges of an optimal solution. In what follows we derive a lower bound for the cost of OPT, using similar ideas to those of [5]. Let $B^i(v,r)$ denote the open ball in G^i with center at v and radius r (we abuse the notation and denote the set of vertices of the ball and the set of edges of the ball by the same notation).

Lemma 15. *Let $r_i = (a,b)$ be a request such that $d^i(a,b) \geq r$. Then, the cost of $OPT \cap B^i(a,r)$ is at least r.*

Proof. In order to serve request r_i, OPT must (at least) pay for a path between a and b in G^i, and the lemma follows from the definitions of $d^i(a,b)$ and G^i. □

A ball $B^i(v,r)$ that satisfies the conditions of Lemma 15 will be referred to as a *lower bound ball* (this term was used in [5] to describe the analogue idea). It follows from Lemma 15 that a collection of pairwise disjoint lower bound balls $\{B^{i_k}(v_k,r_k)\}_k$, comprises a lower bound for OPT (this lower bound is the analogue of the lower bound collection presented by [5]).

Similarly to [5], we will run a shadow algorithm that will find a set of lower bounds for OPT. One of these lower bounds will be sufficient to prove the competitive ratio. For clarity, we use the notation of [5]. For example, $C(k)$ will be a collection of lower bound balls with radius at most 2^k.

The shadow algorithm starts with all $C(k)$'s empty. Upon receiving a request $r_i = (a,b)$ it adds to every $C(j)$ with $j \leq \lfloor \log_2 d^i(a,b) \rfloor$ the largest ball $B^i(a,r)$ such that $r \leq 2^j$ and $B^i(a,r)$ does not intersect any of the previous balls of $C(j)$. The same is performed for b.

Lemma 16. $\Sigma \geq \mathcal{BC}'(\sigma)$.

Proof. We perform an amortized analysis. We create an account for every (A,j) such that $A \in C$ and $j \leq class(A)$. The rest of the analysis is the same as in [5]. Particularly:

- In the first step $\Delta_{cost} = \Delta_\Sigma$;

- If while processing a component $X \in C$ in step 3 (resp. 4) it is the case that $d \geq 2^{m+1}$, then there is no pair of intersecting balls such that one is centered at a (resp. b) and the other is centered at $v \in X$ (pay attention that for $k > i$ it holds that for every $a, b \in V$, $d^i(a,b) \geq d^k(a,b)$ and thus the same proof is valid);
- If while processing a component $X \in C$ in step 3 (resp. 4) it is the case that $d < 2^{m+1}$, then $\Delta_{cost} \leq d - 2^{m+1}$ (exactly same considerations) and $\Delta_\Sigma \geq d - 2^{m+1}$ (note again, that same considerations hold since $k > i$ it holds that for every $a, b \in V$, $d^i(a,b) \geq d^k(a,b)$).

□

Lemma 17. $\max_k \Sigma(k) \geq \frac{1}{\log_2 n+2}\Sigma$.

Proof. By following the same considerations of the proof in the analysis of [5]: removing some lower bound collections at the expense of liquidating the remaining accounts, leaving only $O(\log n)$ collections.. □

Notice that the above analysis implies that the individual efficiency of the strategy vector $((\mathcal{BC}')^{ind}, \ldots, (\mathcal{BC}')^{ind})$ is $O(1)$ away from the best possible. As for the overall efficiency, note that the lower bound on the price of an optimal solution for the induced online problem, that was used to prove the competitive ratio of $\mathcal{BC}'$, is also a lower bound for the price of an optimal solution for the "social" online problem. Hence, the strategy vector $((\mathcal{BC}')^{ind}, \ldots, (\mathcal{BC}')^{ind})$ has a competitive ratio of $O(N \log n)$, which is $O(N)$ away from the known lower bound.

8 Conclusions and Future Research

In this paper we have introduced the model of online games, with the aim of studying scenarios that combine online decision making with interaction between independent agents. As this is the first work in this line of research, we consider it a pioneering work that establishes the foundation for future research.

Our paper has concentrated on equilibria-related issues. A natural question to be considered is what are the limitations of the techniques we have developed? These techniques seem to be ineffective for profit maximization games, such as the game that generalizes the online maximum throughput problem. Moreover, even for the problems we have studied above, it is not clear whether the techniques can be extended and applied to randomized strategies. A negative answer to some of the above questions will strengthen the motivation for the development of other techniques for constructing equilibria. Of special interest is a general technique for the randomized case, as many online problems have randomized online algorithms that outperform the deterministic ones.

A different direction for future research is obtaining negative results of two kinds. The first kind is a lower bound on the competitiveness of strategies which does not follow from a lower bound on an online problem in a straightforward manner, but would rather exploit the nature of interaction between players. A different kind of negative results might deal with the existence of (approximate) equilibria. Another interesting research directions are the price of stability of online games (see Section 2), as well as applying other well-known concepts in algorithmic game theory to the model of online games.

We conclude with a question that we consider to be a by-product of our study. The techniques developed in this paper suggest that an online game induces new online problems having a different flavor than classical online problems. Essentially, in the new problems, in between two requests to an agent, some modifications to the system may occur. Besides the fact that algorithms for these problems may define an equilibrium to the online game, such problems may be of independent interest as they capture certain phenomena that has not been studied so far.

References

[1] Aditya Akella, Srinivasan Seshan, Richard Karp, Scott Shenker, and Christos Papadimitriou, *Selfish behavior and stability of the internet:: a game-theoretic analysis of tcp*, SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications (New York, NY, USA), ACM Press, 2002, pp. 117–130.

[2] Itai Ashlagi, Dov Monderer, and Moshe Tennenholtz, *Resource selection games with unknown number of players*, Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06), 2006.

[3] Baruch Awerbuch, Yossi Azar, and Yair Bartal, *On-line generalized steiner problem*, Proceedings of the 7th annual ACM-SIAM Symposium on Discrete Algorithms (SODA'96) (Philadelphia, PA, USA), Society for Industrial and Applied Mathematics, 1996, pp. 68–74.

[4] Yair Bartal, Amos Fiat, and Yuval Rabani, *Competitive algorithms for distributed data management (extended abstract)*, Proceedings of the 24th annual ACM Symposium on Theory of Computing (STOC'92), ACM Press, May 1992, pp. 39–50.

[5] Piotr Berman and Chris Coulston, *On-line algorithms for steiner tree problems (extended abstract)*, Proceedings of the 29th annual ACM Symposium on Theory of Computing (STOC'97), ACM Press, 1997, pp. 344–353.

[6] Avrim Blum and Jason D. Hartline, *Near-optimal online auctions*, SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (Philadelphia, PA, USA), Society for Industrial and Applied Mathematics, 2005, pp. 1156–1163.

[7] Allan Borodin and Ran El-Yaniv, *Online computation and competitive analysis*, 1st ed., Cambridge University Press, 1998.

[8] Byung-Gon Chun, Kamalika Chaudhuri, Hoeteck Wee, Marco Barreno, Christos H. Papadimitriou, and John Kubiatowicz, *Selfish caching in distributed systems: a game-theoretic analysis*, Proceedings of the 23rd annual ACM Symposium on Principles of Distributed Computing (PODC'04), ACM Press, 2004, pp. 21–30.

[9] Debojyoti Dutta, Ashish Goel, and John Heidemann, *Oblivious aqm and nash equilibria*, SIGCOMM Comput. Commun. Rev. **32** (2002), no. 3, 20–20.

[10] Mohammad T. Hajiaghayi, *Online auctions with re-usable goods*, EC '05: Proceedings of the 6th ACM conference on Electronic commerce (New York, NY, USA), ACM Press, 2005, pp. 165–174.

[11] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes, *Adaptive limited-supply online auctions*, EC '04: Proceedings of the 5th ACM conference on Electronic commerce (New York, NY, USA), ACM Press, 2004, pp. 71–80.

[12] Nathanael Hyafil and Craig Boutilier, *Regret minimizing equilibria and mechanisms for games with strict type uncertainty*, AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence (Arlington, Virginia, United States), AUAI Press, 2004, pp. 268–277.

[13] Makoto Imase and Bernard M. Waxman, *Dynamic steiner tree problem.*, SIAM Journal on Discrete Mathematics **4** (1991), no. 3, 369–384.

[14] Robert D. Kleinberg and Frank Thomson Leighton, *The value of knowing a demand curve: Bounds on regret for online posted-price auctions.*, FOCS, IEEE Computer Society, 2003, pp. 594–605.

[15] Ron Lavi and Noam Nisan, *Competitive analysis of incentive compatible on-line auctions.*, Theor. Comput. Sci. **310** (2004), no. 1-3, 159–180.

[16] ______, *Online ascending auctions for gradually expiring items*, SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (Philadelphia, PA, USA), Society for Industrial and Applied Mathematics, 2005, pp. 1146–1155.

[17] Lavy Libman and Ariel Orda, *Optimal sliding-window strategies in networks with long round-trip delays*, Comput. Networks **46** (2004), no. 2, 219–235.

[18] Boris Teia, *A lower bound for randomized list update algorithms*, Information Processing Letters **47** (1993), no. 1, 5–9.