Scheduling to Maximize Participation*

Ioannis Caragiannis, Christos Kaklamanis, Panagiotis Kanellopoulos, and Evi Papaioannou

Research Academic Computer Technology Institute &
Department of Computer Engineering and Informatics
University of Patras, 26500 Rio, Greece
{caragian,kakl,kanellop,papaioan}@ceid.upatras.gr

Abstract. We study a problem of scheduling client requests to servers. Each client has a particular latency requirement at each server and may choose either to be assigned to some server in order to get serviced provided that her latency requirement is met or not to participate in the assignment at all. From a global perspective, in order to optimize the performance of such a system, one would aim to maximize the number of clients that participate in the assignment. However, clients may behave selfishly in the sense that each of them simply aims to participate in an assignment and get serviced by some server where her latency requirement is met with no regard to the overall system performance. We model this selfish behavior as a strategic game, show how to compute equilibria efficiently, and assess the impact of selfishness on system performance. We also show that the problem of optimizing performance is computationally hard to solve, even in a coordinated way, and present efficient approximation and online algorithms.

1 Introduction

We are motivated by the following scenario where clients aim to retrieve some objects (e.g., video/audio files) from servers (each server can be thought of as an electronic store). Each client requests one object which may exist in some of the servers. In order to get serviced, the client has to connect to the server and download the object. The service time (or latency) for a client connected to a server is proportional to the number of simultaneous connections to that server. Clients may value differently the service received from each server in the sense that if the latency is high enough, the client may decide not to receive the object from that server and close the connection. A client may connect to a server and download the requested object if the current load of the server is within her valuation criterion; of course, this action could regret some other client connected to the server who will decide not to get serviced by that server and will make another choice. A client may decide not to get serviced at all if no server meets her valuation criterion.

* This work was partially supported by the European Union under IST FET Integrated Project 015964 AEOLUS.

U. Montanari, D. Sannella, and R. Bruni (Eds.): TGC 2006, LNCS 4661, pp. 218–232, 2007.

Naturally, such a scenario of selfish behavior can be modeled using the notion of a strategic game from game theory. We define a particular class of games called SMP games. In an SMP game, we have a set of clients C and a set of servers M. Each client $c \in C$ has a non-negative finite integer latency bound ℓ_c^k at each server $k \in M$. Clients are non-cooperative in the sense that each client wishes to be assigned to some server where her latency bound is satisfied; otherwise, she prefers not to be assigned to any server. Given an assignment of some of the clients to servers, an SMP game is defined by the following payoff function for each client: a client assigned to a server k together with $n_k - 1$ other clients has payoff 1 if $\ell_c^k \geq n_k$ and payoff -1 otherwise. A client that is not assigned to any server has zero payoff. We say that an assignment is *valid* if no client has payoff -1. An assignment is a *pure Nash equilibrium* (or, simply, an *equilibrium*) for an SMP game if no client has an incentive to unilaterally change her strategy. Clearly, any equilibrium is a valid assignment. The benefit of an assignment for an SMP game is the sum of the payoffs of all clients. Hence, the benefit of a valid assignment equals the number of clients accommodated in servers. We use the notion of the *price of anarchy* introduced in [20] (see also [16]) to assess the quality of equilibria of SMP games. The price of anarchy of an SMP game is defined as the ratio of the benefit of an optimal assignment for the SMP game over the benefit of the worst equilibrium.

Selfish behavior could be bypassed by having a *scheduler* which knows the load of each of the available servers, receives the requests of the clients for objects together with their latency bound for each server, and coordinates the assignment of clients to servers. Although such an approach would be inappropriate and unrealistic in a networked environment, it is important to consider it in order to compare it with the uncoordinated case and assess selfish behavior. Furthermore, it gives rise to an interesting combinatorial optimization problem which we call *scheduling to maximize participation* (SMP). Although scheduling optimization problems (focusing mainly on minimizing some function of the server latencies when all clients must be assigned to a server, e.g., see [17]) and corresponding games (such as load balancing [6,8,12,13,15,16,18] and congestion games [1,7,21]) have been extensively studied in the literature, to the best of our knowledge, SMP has not been studied before.

Our main motivating question concerns the efficient construction of equilibria in SMP games. In Section 2, we first present a Nashification technique which, starting from a valid assignment, converges to an equilibrium by making a polynomial number of client moves. This is motivated by [12] where a similar in spirit technique has been applied to a different scheduling game. We also show that the price of anarchy of any SMP game is at most 2. Hence, the Nashification technique provides an algorithm for approximating both the best and the worst equilibrium within a factor of 2; these two problems are proved to be APX-hard. An important property of our Nashification technique is that the benefit of the equilibrium computed is not smaller than the benefit of the initial assignment. So, in order to compute equilibria of large benefit, it suffices to compute valid assignments of large benefit. We use the term SMP to refer to the optimization

problem that corresponds to the problem of computing a valid assignment of maximum benefit.

In Section 3, we present a $\frac{e}{e-1} \approx 1.58$-approximation SMP algorithm based on linear programming and randomized rounding. SMP can be thought of as a special case of *combinatorial auctions*. In the problem of combinatorial auctions, we have a set of players and a set of items. A feasible allocation assigns every item to at most one player. For every player, her utility function w_i is defined over the set of items that she receives. The goal is to find a feasible allocation that maximizes social welfare $\sum_i w_i(S_i)$, where S_i is the set of items allocated to player i. In SMP, the servers correspond to players and the clients correspond to items; the utility of a set of clients for a server is the maximum number of clients from this set that can be assigned to the server in any valid assignment. Recent work [9,10,11] presents $\frac{e}{e-1}$-approximation algorithms (also based on linear programming and randomized rounding) for combinatorial auctions when the utility functions have special properties (e.g., subadditivity). However, these techniques are rather impractical since they make use of the ellipsoid method in order to solve the corresponding linear programming relaxation which is of exponential size. The LP corresponding to SMP has also an exponential number of variables. In this paper, for SMP, we exploit the structure of the problem and prove that SMP is equivalent to a constrained integral flow problem whose LP relaxation is of polynomial size and, hence, it can be solved by practical linear programming algorithms. Then, randomized rounding in our case is simpler compared to [9,10,11]. On the negative side, we show that the problem is APX-hard by showing an explicit inapproximability result of 368/367 using a reduction from multidimensional matching (this result also implies the APX-hardness of the problem of computing the best equilibrium for an SMP game).

In Section 4, we consider the online version of SMP where clients (together with their latency bound vector) appear online and an irrevocable decision has to be made when each client c appears. This means that client c can either be rejected or put in a server so that neither the latency bounds of previously assigned clients nor the latency bound of c herself is violated. This can be thought of as the problem of computing an efficient valid assignment for an SMP game when information about the game is gradually revealed to the algorithm. Here, we assess the quality of the solution in terms of the *competitive ratio* [4] (or *competitiveness*) defined as the maximum over all SMP sequences of the ratio of the optimal benefit over the expected benefit of the assignment computed by the algorithm. In general, we assume that sequences are generated by oblivious adversaries which may have knowledge of the probability distribution that may be used by the algorithm but have no access to its random choices (if any). The online version is inherently more difficult to approximate since, as we prove, no deterministic algorithm is better than T-competitive while no randomized algorithm can be better than H_T-competitive against oblivious adversaries, where T is the ratio of the maximum over the minimum non-zero latency bound over all clients and servers and H is the harmonic function. On the positive side, we show an asymptotically tight upper bound by presenting an $O(\ln T)$-competitive

randomized algorithm that needs to know T in advance. In the case where no information about T is known in advance, a slightly inferior competitiveness bound is obtained. Our online algorithms are based on the *classify-and-randomly-select* paradigm (see [4]) which has been proved to be useful in other problems (e.g., call admission control in communication networks [2,3], online independent set problems [5], etc.).

2 Equilibria and Price of Anarchy

We begin by presenting our Nashification technique. We describe algorithm Nashify which starts from a valid assignment for an SMP game consisting of a set C of n clients and a set M of m servers and works as follows. It proceeds in rounds. Denote by n_k^i the number of clients assigned to server k at the beginning of round i. In each round, algorithm Nashify performs one step for each server $k \in M$. In each step of a round i, at most two moves are performed. If there exists a client c not assigned to any server which has latency bound $l_c^k > n_k^i$, c moves to server k. If there exists another client c' in server k with $l_{c'}^k = n_k^i$ (i.e., her latency bound is violated by the move of client c), then client c' moves out of server k (and is not assigned to any server). Algorithm Nashify terminates when no move is performed during a whole round. We prove the following statement.

Theorem 1 (Nashification). *For any* SMP *game with n clients and m servers, algorithm* Nashify *computes an equilibrium of benefit not smaller than the benefit of the initial assignment by performing at most $2nm$ moves.*

Proof. The assignment produced by algorithm Nashify is valid since the condition that the number of clients in any server is not greater than the latency bound of each client assigned to this server remains true after each step of the algorithm. It is also an equilibrium since, by the termination criterion, no unassigned client has an incentive to move to any server. Furthermore, during a step, a client may move out of a server only if another client moves to this server. Hence, the benefit of the final assignment is not smaller than the benefit of the initial one. In order to prove the upper bound on the total number of moves, observe that a client c that moves out of a server k at some round i has $l_c^k = n_k^i$. Since the number of clients at server k never decreases in later rounds, client c will never move to server k again. So, the total number of moves each of the n clients can make is at most $2m$ (one move in and one move out for each of the m servers). □

Implicitly, in the proof of Theorem 1 we also prove that SMP games always have equilibria. The next result states that their benefit is fairly large.

Theorem 2. *The price of anarchy of any* SMP *game is at most* 2.

Proof. Consider an equilibrium for an SMP game on a set of servers M and an optimal assignment. We denote by o_j the number of clients that are in server j in the optimal assignment. Let n_j be the number of clients in server j in the equilibrium and R_j be the set of clients that are in server j in the optimal

assignment but are not assigned to any server in the equilibrium. Consider a client $c \in R_j$ for some server j. Since c is in server j in the optimal assignment, it holds that $\ell_c^j \geq o_j$. Since c is not assigned to any server in the equilibrium, it holds that $\ell_c^j \leq n_j$, otherwise c would have an incentive to move to server j. Thus, $o_j \leq n_j$ for any server j for which $R_j \neq \emptyset$. It holds that

$$\sum_{j \in M} o_j = \sum_{j \in M: R_j = \emptyset} o_j + \sum_{j \in M: R_j \neq \emptyset} o_j \leq \sum_{j \in M: R_j = \emptyset} o_j + \sum_{j \in M: R_j \neq \emptyset} n_j$$
$$\leq \sum_{j \in M} n_j + \sum_{j \in M: R_j \neq \emptyset} n_j \leq 2 \sum_{j \in M} n_j \qquad \square$$

The above result is tight since there exists a simple matching lower bound consisting of two servers a, b and two clients x, y with $\ell_x^a = \ell_x^b = \ell_y^a = 1$ and $\ell_y^b = 0$. In the optimal solution, x is assigned to b and y is assigned to a, while the assignment where x is assigned to a and y is not assigned to any server is an equilibrium.

Algorithm Nashify is essentially a 2–approximation algorithm for computing the best equilibrium for SMP games. Starting from any initial valid assignment for an SMP game, it computes an equilibrium which (by Theorem 2) has benefit at least half the optimal benefit. In Section 3, we present an algorithm that computes a valid assignment of benefit at most 1.58 times smaller than the optimal benefit. Combined with algorithm Nashify, this yields a 1.58–approximation algorithm for computing the best equilibrium for SMP games. Clearly, algorithm Nashify is also a 2–approximation algorithm for computing the worst equilibrium for SMP games.

Concerning the hardness of approximation of the problem of computing the best equilibrium for SMP games, this follows by a statement in the next section where we show that the problem of computing the best valid assignment is APX-hard. The next theorem shows that the problem of computing the worst equilibrium is APX-hard as well.

Theorem 3. *The problem of computing the worst equilibrium for* SMP *games is APX-hard.*

Proof. We will show that there are instances of the problem which are equivalent to instances of Minimum Maximal Bipartite Matching which is known to be APX-hard [22]. An instance of Minimum Maximal Bipartite Matching consists of a bipartite graph $G(U, V, E)$ and the objective is to compute a maximal matching of minimum size. A matching is called maximal if we cannot obtain another matching by adding one extra edge to it. Consider such an instance consisting of a bipartite graph $G(U, V, E)$ and construct an SMP game consisting of a client for each node of U and a server for each node of V. The latency bound of a client corresponding to a node $u \in U$ is 1 to each server corresponding to a node $v \in V$ such that $(u, v) \in E$ and 0 otherwise. We will show that there exists a maximal matching in G of size K if and only if the SMP game has an equilibrium of benefit K. Consider a maximal matching in

G consisting of K edges. Then, an equilibrium for the SMP game is defined as follows. For each edge (u, v) in the matching, the client corresponding to node u is assigned to the server corresponding to node v. The fact that M is a matching guarantees that each client is assigned to at most one server and each server receives at most one client, i.e., the assignment is valid. In order to prove that it is also an equilibrium, observe that since the matching is maximal there is no edge $(u, v) \in E \backslash M$ such that neither u nor v is an endpoint of an edge in M. Hence, no client that is not assigned to any server has an incentive to move to some server. Similarly, consider an equilibrium for the SMP game. Each client is assigned to at most one server and, by the definition of the latency bounds, each server contains at most one client. Hence, the set consisting of the edges (u, v) where $v \in V$ corresponds to a server containing the client corresponding to node $u \in U$ is a matching in G. Since the assignment is an equilibrium, no client c among those not assigned to any server has an incentive to move to some server k such that $\ell_c^k = 1$ because server k already contains some other client. This implies that all edges in $E \backslash M$ are adjacent to some edge in M which means that M is maximal. □

3 Computation of Efficient Valid Assignments

In this section, we present an algorithm that computes valid assignments for SMP instances. The algorithm is based on linear programming and randomized rounding and achieves an approximation guarantee of $\frac{e}{e-1} \approx 1.58$.

Given an instance of SMP consisting of a set C of n clients and a set M of m servers, we say that a set of clients $A \subseteq C$ is valid for server $k \in M$ if $\ell_c^k \geq |A|$ for each client $c \in A$. Hence, a valid set for server k is a set of clients which can be accommodated in server k in a solution of SMP. Now, SMP is to select one valid set of clients for each server so that the valid sets selected are disjoint and the total number of clients in valid sets selected is maximized. This is equivalent to the following integer linear program:

$$
\begin{aligned}
\text{(ILP)} \quad \text{maximize} \quad & \sum_{k \in M} \sum_{A \in \mathcal{A}_k} x_A^k |A| \\
\text{subject to} \quad & \sum_{k \in M} \sum_{A \in \mathcal{A}_k : c \in A} x_A^k \leq 1, \text{ for any } c \in C \\
& \sum_{A \in \mathcal{A}_k} x_A^k \leq 1, \text{ for any } k \in M \\
& x_A^k \in \{0, 1\}, \text{ for any } k \in M \text{ and } A \in \mathcal{A}_k.
\end{aligned}
$$

where $\mathcal{A}$ denotes the set of all valid sets and $\mathcal{A}_k$ denotes the set of valid sets for server k. The variable x_A^k denotes whether the valid set A is selected at server k. The constraints guarantee that each client is assigned to at most one server (i.e., it belongs to at most one valid set) and that at most one valid set is selected for each server.

We will first show that (ILP) is equivalent to a constrained integral flow problem in a particular polynomially sized network N presented in the following. For each server $k = 1, ..., m$ the network has three nodes s_k (called the source node associated with server k), s'_k, and t_k (called the sink node associated with server k) and, for each $i = 1, ..., n$, it has two nodes u_i^k and v_i^k. For each server $k = 1, ..., m$, node s_k is connected through a directed link to node s'_k, and for each $i = 1, ..., n$, node s'_k is connected through a directed link to node u_i^k, while node v_i^k is connected through a directed link to node t_k. For each server $k = 1, ..., m$, each $i = 1, ..., n$, each $j = 1, ..., i$, and each client $c \in C$, the network has two nodes $w_i^k(c, j)$ and $z_i^k(c, j)$. For each server $k = 1, ..., m$ and each $i = 1, ..., n$, node u_i^k is connected through a directed link to all nodes $w_i^k(c, 1)$ for each client c such that $\ell_c^k \geq i$. All nodes $z_i^k(c, i)$ are connected through a directed link to node v_i^k. For each server $k = 1, ..., m$, each $i = 1, ..., n$, and each $j = 1, ..., i$, node $w_i^k(c, j)$ is connected to node $z_i^k(c, j)$. For each server $k = 1, ..., m$, we fix an ordering π_k of the clients in C (i.e., π_k assigns a distinct integer in $\{1, 2, ..., n\}$ to each client) such that $\pi_k(c') > \pi_k(c)$ implies $\ell_{c'}^k \geq \ell_c^k$. For each server $k = 1, ..., m$, each $i = 2, ..., n$, and each $j = 1, ..., i-1$, node $z_i^k(c, j)$ is connected to nodes $w_i^k(c', j+1)$ for all clients c' with $\pi_k(c') > \pi_k(c)$. All edges have unit capacity. The edge connecting nodes $w_i^k(c, j)$ and $z_i^k(c, j)$ (for some client $c \in C$, each server $k = 1, ..., m$, each $i = 1, ..., n$, and each $j = 1, ..., i$) belongs to the edge-set E_c of client c. Such edges are called *client edges*. An example of this construction is presented in Figure 1.

We observe that there is an $1 - 1$ correspondence between valid sets of clients and source-sink paths in N. Indeed, consider a valid set of clients $A = \{c_1, c_2, ..., c_i\}$ for server k and assume without loss of generality that $\pi_k(c_j) < \pi_k(c_{j+1})$, for $j = 1, ..., i-1$. Then, nodes s_k and t_k are connected through the path

$$\left\langle s_k, s'_k, u_i^k, w_i^k(c_1, 1), z_i^k(c_1, 1), w_i^k(c_2, 2), z_i^k(c_2, 2), ..., w_i^k(c_i, i), z_i^k(c_i, i), v_i^k, t_k \right\rangle.$$

Similarly, consider a path p from node s_k to node t_k in network N. By the definition of the network, the path consists of the subpath $\left\langle s_k, s'_k, u_i^k \right\rangle$, a subpath

$$p' = \left\langle u_i^k, w_i^k(c_1, 1), z_i^k(c_1, 1), w_i^k(c_2, 2), z_i^k(c_2, 2), ..., w_i^k(c_i, i), z_i^k(c_i, i), v_i^k \right\rangle$$

connecting node u_i^k to node v_i^k, and the subpath $\left\langle v_i^k, t_k \right\rangle$ for some i. The client edges in subpath p' belong to different clients since an edge connecting node $z_i^k(c_j, j)$ to $w_i^k(c_{j+1}, j+1)$ implies that $\pi_k(c_j) < \pi_k(c_{j+1})$. Consequently, it holds that $\ell_{c_j}^k \leq \ell_{c_{j+1}}^k$ for $j = 1, ..., i-1$. Furthermore, an edge connecting node u_i^k to $w_i^k(c_1, 1)$ implies that $\ell_{c_1}^k \geq i$. So, it holds that $\ell_{c_j}^k \geq i$ for $j = 1, ..., i$, i.e., the set of clients $\{c_1, c_2, ..., c_i\}$ is valid.

Now, SMP can be thought of as the following constrained integral flow problem: the objective is to push flow f from the source nodes to the sink nodes so that the flow f_e carried by each edge e is integral, the capacity constraints are maintained (i.e., $f_e \leq 1$ for each edge e), the total flow carried by all edges in the edge-set E_c is at most 1 for each client $c \in C$, and the quantity

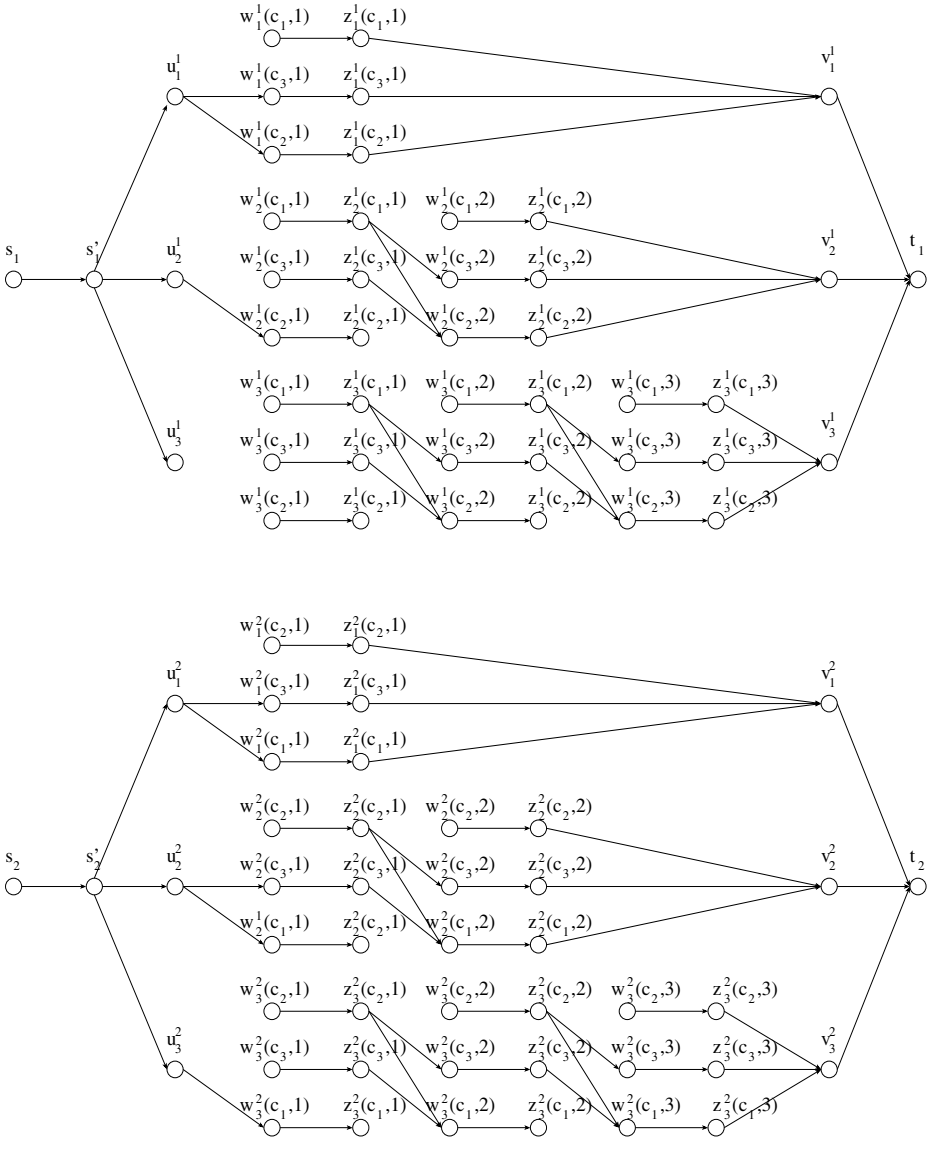

Fig. 1. The corresponding network constructed by the algorithm of Section 3 for an instance of SMP with two servers 1 and 2 and three clients c_1, c_2, c_3 with latency bounds $\ell^1_{c_1} = 0$, $\ell^2_{c_1} = 3$, $\ell^1_{c_2} = 2$, $\ell^2_{c_2} = 0$, $\ell^1_{c_3} = 1$, and $\ell^2_{c_3} = 2$. The ordering in each server is defined as $\pi_1(c_1) = 1$, $\pi_1(c_2) = 3$, $\pi_1(c_3) = 2$ and $\pi_2(c_1) = 3$, $\pi_2(c_2) = 1$, $\pi_2(c_3) = 2$. An optimal solution to the corresponding constrained integral flow problem consists of the paths $\langle s_1, s'_1, u^1_1, w^1_1(c_2, 1), z^1_1(c_2, 1), v^1_1, t_1\rangle$ and $\langle s_2, s'_2, u^2_2, w^2_2(c_3, 1), z^2_2(c_3, 1), w^2_2(c_1, 2), z^2_2(c_1, 2), v^2_2, t_2\rangle$ corresponding to the assignment of client c_2 to server 1 and clients c_1 and c_3 to server 2.

$\sum_{k \in M} \sum_{i=1}^{n} i \cdot f_{(v_i^k, t_k)}$ is maximized. The constraints imply that the solution to the constrained integral flow problem will consist of at most one path connecting each source node to its corresponding sink node so that the client edges in these paths belong to different clients. The quantity to be maximized equals the number of client edges that carry some flow. Equivalently, we obtain at most one valid set of clients per server so that the valid sets are disjoint and contain a maximum number of clients for the original SMP instance.

Since, as we show later in this section, SMP is APX-hard, we cannot hope to solve optimally the constrained integral flow problem. Instead, we relax the integrality constraint and solve the corresponding constrained fractional flow problem by transforming it to a linear program. Here, the variables of the linear program represent the flow f_e carried by each edge e and the constraints of the linear program are either flow conservation constraints at the network nodes, or capacity constraints at the network edges, or require that the total flow carried by the edges of the edge-set of any client is at most 1. Note that, although (ILP) has an exponential number of variables, the constrained fractional flow problem can be expressed as a linear program with polynomial number of variables and constraints since the network constructed has at most $O(n^3 m)$ nodes and $O(n^4 m)$ edges.

Once we have a solution to the constrained fractional flow problem, we can obtain a solution to the linear programming relaxation of (ILP) obtained by relaxing the integrality constraint to $x_A^k \geq 0$. This can be done by decomposing the flow into flow paths using a folklore *path stripping* technique. For $k = 1, ..., m$, we pick the edge e carrying the smallest non-zero amount of flow between nodes s_k and t_k and compute a path p from s_k to t_k that contains e and consists of edges carrying non-zero amounts of flow. We set the flow carried by the flow path p to $\hat{f}_p = f_e$ and decrease the flow on each edge in p by f_e. We repeat this procedure and decompose all flow between nodes s_k and t_k into flow paths. Note that the number of paths obtained in this way is not greater than the number of edges in the network since, in each step, the flow variable of some edge is decreased to zero. After performing path stripping, we obtain a fractional solution to the linear programming relaxation of (ILP) by setting $x_A^k = \hat{f}_p$ for each valid set of clients A corresponding to a flow path p carrying a non-zero amount of flow $\hat{f}_p$ between nodes s_k and t_k and implicitly setting all other variables to 0.

In order to obtain an integral feasible solution to (ILP), we will use randomized rounding. Due to the special structure of SMP, randomized rounding and its analysis are simpler compared to [9,10,11]. We cast a die for each server k having one face for each valid set $A \in \mathcal{A}_k$ with $x_A^k > 0$ (with probability that this face is the outcome of the die-casting equal to x_A^k) and an additional face corresponding to the fact that no client is accepted at server k (with the probability that this face is the outcome of the die-casting equal to $1 - \sum_{A \in \mathcal{A}_k} x_A^k$). After performing the die-castings for all servers, we perform a correction procedure by assigning each client c to that server k (if any) where a set containing client c is the outcome of the die-casting for server k; if more than one die-castings have outcomes containing the same client, then client c is assigned to one of the

corresponding servers arbitrarily. The assignment produced is valid since the first set of constraints of (ILP) is guaranteed by the correction procedure while the second set of constraints is guaranteed by randomized rounding. Clearly, all sets produced by the correction procedure are valid since removing a client from a valid set still gives a valid set of clients.

Lemma 1. *Given an instance of* SMP*, the algorithm computes a valid assignment with expected benefit at least* $1 - \frac{1}{e}$ *times the optimal benefit.*

Proof. Denote by Y_c the 0/1 random variable denoting whether client c is contained in some of the valid sets selected after the application of randomized rounding. The probability that a client is contained in some of the valid sets for server k selected after the randomized rounding is $\sum_{A \in \mathcal{A}_k : c \in A} x_A^k$ and, since die-castings are performed independently, the probability that a client $c \in C$ is contained in some of the valid sets selected after the randomized rounding is

$$\Pr[Y_c = 1] = 1 - \prod_{k \in M} \left(1 - \sum_{A \in \mathcal{A}_k : c \in A} x_A^k \right) \geq 1 - \exp\left(- \sum_{k \in M} \sum_{A \in \mathcal{A}_k : c \in A} x_A^k \right)$$

where the inequality follows since $\prod_{i=1}^{n} (1 - x_i) \leq \exp\left(-\sum_{i=1}^{n} x_i\right)$ when $x_i \in [0, 1]$.

Denote by $\hat{x}_A^k$ the solution obtained after the application of the correction procedure. Since a client that is contained in some valid set selected by the randomized rounding procedure also appears in exactly one valid set after the correction procedure, the benefit of the final solution is $\sum_{k \in M} \sum_{A \in \mathcal{A}_k} \hat{x}_A^k |A| = \sum_{c \in C} Y_c$. Hence, we obtain that the expected benefit of the final solution is

$$\begin{aligned}
\mathcal{E}\left[\sum_{k \in M} \sum_{A \in \mathcal{A}_k} \hat{x}_A^k |A| \right] &= \mathcal{E}\left[\sum_{c \in C} Y_c \right] \\
&= \sum_{c \in C} \Pr[Y_c = 1] \\
&\geq \sum_{c \in C} \left(1 - \exp\left(- \sum_{k \in M} \sum_{A \in \mathcal{A}_k : c \in A} x_A^k \right) \right) \\
&\geq \sum_{c \in C} \left(1 - e^{-1}\right) \sum_{k \in M} \sum_{A \in \mathcal{A}_k : c \in A} x_A^k \\
&= \left(1 - e^{-1}\right) \sum_{c \in C} \sum_{k \in M} \sum_{A \in \mathcal{A}_k : c \in A} x_A^k \\
&= \left(1 - e^{-1}\right) \sum_{k \in M} \sum_{A \in \mathcal{A}_k} x_A^k |A| \\
&\geq \left(1 - e^{-1}\right) \sum_{k \in M} \sum_{A \in \mathcal{A}_k} x_A^{*k} |A|
\end{aligned}$$

where x^* denotes the optimal integral SMP solution. The second inequality follows since $1 - \exp(-x) \geq (1 - e^{-1})x$ when $x \in [0, 1]$ and due to the constraint of the linear program. □

The algorithm can be forced to obtain a ratio which is within any constant $\epsilon > 0$ close to $\frac{e}{e-1}$ with high probability by applying the randomized rounding procedure $O(1/\epsilon)$ times; this follows by a simple application of the Markov inequality [19]. The next statement summarizes the discussion in this section.

Theorem 4. *There exists an* $\frac{e}{e-1} \approx 1.58$*-approximation algorithm for* SMP.

On the negative side, we show that SMP is APX-hard.

Theorem 5. *For any* $\epsilon > 0$*, it is NP-hard to approximate* SMP *within* $368/367-\epsilon$.

Proof. We use a reduction from 6-dimensional matching which is known to be APX-hard. An instance of 6-dimensional matching consists of a 6-uniform 6-partite hypergraph G and the objective is to compute a matching of maximum size in G (i.e., a set of hyperedges of maximum cardinality in which no two of them share any node). In particular, [14] shows that there exist instances of 6-dimensional matching consisting of a 6-uniform 6-partite hypergraph with n nodes and $n/2$ edges for which, for any $\epsilon \in (0, 1/46)$, it is NP-hard to decide whether the maximum matching has size at least $(1-\epsilon)\frac{n}{6}$ or at most $\left(\frac{22}{23}+\epsilon\right)\frac{n}{6}$.

Given such an instance I_{6DM} of 6-dimensional matching consisting of a hypergraph G, we construct the instance I_{SMP} of SMP that contains a server for each hyperedge of G, a client for each node of G and $\frac{5n}{2}$ additional clients. Each of the additional clients has latency bound 5 at all servers while the client corresponding to node v of G has latency bound 6 at each server corresponding to hyperedges of G containing v and latency bound 5 at all other servers.

We say that a solution to I_{SMP} is maximal if it contains at least 5 clients per server. Note that given a solution to I_{SMP}, we can compute a maximal solution of at least the same benefit by accommodating additional clients to the servers that contain less than 5 clients. We observe that any solution for I_{6DM} of cardinality K can be converted to a maximal solution for I_{SMP} of benefit $\frac{5n}{2}+K$ by assigning to each server corresponding to a hyperedge in the solution of I_{6DM} the clients corresponding to nodes contained in the hyperedge and 5 of the additional clients to any other server. Similarly, any maximal solution for I_{SMP} of benefit $\frac{5n}{2}+K$ can be converted to a solution for I_{6DM} of cardinality K by simply considering the hyperedges corresponding to servers with 6 clients. Hence, if for some $\epsilon \in (0, 1/46)$ we could decide whether the optimal benefit for I_{SMP} is above $\left(\frac{368}{23}-\epsilon\right)\frac{n}{6}$ or below $\left(\frac{367}{23}+\epsilon\right)\frac{n}{6}$ then we could decide whether the maximum matching in I_{6DM} has cardinality at least $\left(\frac{368}{23}-\epsilon\right)\frac{n}{6}-\frac{5n}{2} = (1-\epsilon)\frac{n}{6}$ or at most $\left(\frac{367}{23}+\epsilon\right)\frac{n}{6}-\frac{5n}{2} = \left(\frac{22}{23}+\epsilon\right)\frac{n}{6}$. □

4 Online Algorithms

In this section we consider the online version of SMP. Observe that deterministic online algorithms for SMP are at least T-competitive, where T is the ratio of the maximum over the minimum latency bound over all clients. To see this, consider an instance of SMP where a single server is available, a deterministic algorithm

A and an offline adversary $\mathcal{ADV}$ working as follows. First, the adversary presents one client of latency bound 1. If the algorithm A rejects the client, the adversary stops the sequence; in this case A has no benefit. Otherwise (i.e., if A accepts the client of latency bound 1), the adversary presents T clients each with latency bound T. In this case, the benefit of the algorithm A is 1, while the optimal benefit is T.

In what follows, using Yao's Minimax Principle (see [4,19]), we prove a lower bound on the competitive ratio of any randomized online SMP algorithm against oblivious adversaries. In our proof, we use the following lemma.

Lemma 2 (Minimax Principle [4,19]). *Given a probability distribution $\mathcal{P}$ over sequences of clients, denote by $\mathcal{E}_{\mathcal{P}}[B_A]$ and $\mathcal{E}_{\mathcal{P}}[B_{OPT}]$ the expected benefit of a deterministic algorithm A and the expected optimal benefit on sequences of clients generated according to $\mathcal{P}$. Define the competitiveness $c_A^{\mathcal{P}}$ of A under $\mathcal{P}$ to be*

$$c_A^{\mathcal{P}} = \frac{\mathcal{E}_{\mathcal{P}}[B_{OPT}]}{\mathcal{E}_{\mathcal{P}}[B_A]}$$

and let c be the minimum of $c_A^{\mathcal{P}}$ over all deterministic algorithms A. Then, c is a lower bound on the competitiveness of any randomized algorithm A_R against an oblivious adversary.

Our lower bound is the following.

Theorem 6. *Any (possibly randomized) online* SMP *algorithm has competitive ratio at least H_T against oblivious adversaries, where T is the ratio of the maximum over the minimum latency bound over all clients and servers.*

Proof. We will prove that there exists an adversary $\mathcal{ADV}$ that presents sequences of clients with latency bounds between 1 and T according to a probability distribution $\mathcal{P}$ in such way that no deterministic algorithm can be better than H_T–competitive under $\mathcal{P}$. Then, the theorem will follow by Lemma 2.

The adversary $\mathcal{ADV}$ runs at most T phases at a single server. At phase i, $1 \leq i \leq T$, it presents i clients of latency bound i. The adversary $\mathcal{ADV}$ starts with phase 1. After running phase i with $1 \leq i \leq T-1$, $\mathcal{ADV}$ tosses a coin with probability Pr[heads] $= \frac{1}{i+1}$. On heads, it stops the sequence; on tails, it proceeds to phase $i+1$. After having run phase T, the adversary $\mathcal{ADV}$ stops the sequence.

Consider an algorithm and assume that the first client it accepts belongs to phase i. Since the latency bound of this client is i, the algorithm cannot accept more than $i-1$ additional clients. So, the best the algorithm can do is to accept all clients of phase i. Thus, in order to prove the lower bound, it suffices to consider the deterministic algorithm A_t (for $t = 1, ..., T$) that waits for the first $t-1$ phases of the sequence accepting no clients and (if phase t is run) accepts all t clients of phase t. Clearly, algorithm A_1 has benefit 1. The probability that the adversary runs phase t is $\frac{1}{t}$ (which is the probability that the adversary $\mathcal{ADV}$ continues after each of the first $t-1$ phases). So, A_t has benefit t with

probability $\frac{1}{t}$ and no benefit with probability $1-\frac{1}{t}$. Hence, the expected benefit of A_t under $\mathcal{P}$ is 1.

The optimal benefit for a sequence produced by $\mathcal{ADV}$ is obtained by accepting all clients of the last phase of the sequence. Denote by p_i the probability that phase i is the last phase the adversary runs. It is $p_T = \frac{1}{T}$ while for $i = 1, ..., T-1$ it is $p_i = \frac{1}{i(i+1)}$. We obtain that the expected optimal benefit under $\mathcal{P}$ is $\mathcal{E}_{\mathcal{P}}[B_{OPT}] = \sum_{i=1}^{T} i \cdot p_i = \sum_{i=1}^{T} \frac{1}{i} = H_T$. □

In the following, we present the randomized online SMP algorithm Classify. Assume that the algorithm knows in advance the values $\ell_{\min}$ and $\ell_{\max}$ of the minimum and maximum non-zero latency bounds of any client of the sequence at any server. Let $T = \ell_{\max}/\ell_{\min}$. The algorithm uses a parameter γ (to be defined later) and equiprobably selects an integer i from 0 to $\lceil \log_\gamma T \rceil - 1$. When a client appears, the algorithm checks whether her latency bound at some server is in the interval $\left[\ell_{\min}\gamma^i, \ell_{\min}\gamma^{i+1}\right)$ and whether assigning the client to this server is feasible in the sense that neither the latency bound of previously accepted clients at this server nor the latency bound of the client herself at this server are violated. If such a server exists, the algorithm assigns the client to this particular server (ties are broken arbitrarily); otherwise, it rejects the client. We prove the following theorem.

Theorem 7. *Algorithm* Classify *has competitive ratio at most* $1 + \gamma + \frac{1+\gamma}{\ln \gamma} \ln T$ *against oblivious adversaries.*

Proof. Denote by OPT the optimal set of clients of the sequence. We define a partition of OPT in $\lceil \log_\gamma T \rceil$ disjoint subsets OPT_i of OPT for $i = 0, ..., \lceil \log_\gamma T \rceil - 1$. For each client c and integer $i = 0, ..., \lceil \log_\gamma T \rceil - 1$, denote by F_c^i the set of servers at which the client c has latency bound in the interval $[\ell_{\min}\gamma^i, \ell_{\min}\gamma^{i+1})$. A client c belongs to OPT_i if it is accepted at a server in F_c^i in the optimal solution.

Assume that algorithm Classify has selected integer i. Then, it considers the original sequence as a new sequence σ_i where each client c has latency bound $\ell_c'^k = \ell_c^k$ if $k \in F_c^i$ and $\ell_c'^k = 0$, otherwise. Denote by O_i the optimal set of clients for σ_i. By the definition of the sequence σ_i and the set OPT_i, it is $|O_i| \geq |OPT_i|$.

First, we show that the benefit B_i of the algorithm Classify when it selects integer i is $B_i \geq \frac{1}{\gamma+1}|O_i| \geq \frac{1}{\gamma+1}|OPT_i|$. Denote by A and R the sets of clients accepted and rejected, respectively, by the algorithm Classify when it selects integer i. For each server k, denote by A_k the set of clients accepted at server k by algorithm Classify and by O_i^k the set of clients accepted at server k in the optimal solution for σ_i. Since the latency bound of any client in A_k at server k is at most γ times smaller than the latency bound of any client in $R \cap O_i^k$ at server k and since no client from $R \cap O_i^k$ can fit in server k, it holds that $|A_k| \geq \frac{1}{\gamma}|R \cap O_i^k|$. So, for the benefit B_i we have

$$B_i = |A| \geq \frac{1}{\gamma+1}|A| + \frac{\gamma}{\gamma+1}\sum_k |A_k| \geq \frac{1}{\gamma+1}|A \cap O_i| + \frac{1}{\gamma+1}\sum_k |R \cap O_i^k|$$

$$= \frac{1}{\gamma+1}\left(|A \cap O_i| + |R \cap O_i|\right) = \frac{1}{\gamma+1}|O_i| \geq \frac{1}{\gamma+1}|OPT_i|.$$

Now, by linearity of expectation, we obtain that the expected benefit of the algorithm is

$$E[B] = \sum_{i=0}^{\lceil \log_\gamma T \rceil - 1} (\Pr[i \text{ is selected}] \cdot B_i) \geq \frac{1}{(1+\gamma)\lceil \log_\gamma T \rceil} \sum_{i=0}^{\lceil \log_\gamma T \rceil - 1} |OPT_i|$$
$$\geq \frac{1}{(1+\gamma)\left(1+\log_\gamma T\right)} |OPT|.$$

Hence, the competitive ratio of the algorithm is $1 + \gamma + \frac{1+\gamma}{\ln \gamma} \ln T$. □

The expression in Theorem 7 is minimized to approximately $4.6 + 3.59 \ln T$ for $\gamma = 3.6$. Note that we have assumed that algorithm Classify knows the maximum and minimum over the non-zero latency bounds of all clients at all servers $\ell_{\max}$ and $\ell_{\min}$ in advance (and, consequently, it knows their ratio T). If it only knows T, when the first client appears, it may assume $\ell_{\max} = \ell T$ and $\ell_{\min} = \max\{1, \ell/T\}$ where ℓ is any non-zero latency bound of the first client appeared at some server. Then, the analysis proceeds along very similar lines to the proof of Theorem 7 and yields a competitive ratio only a constant factor worse than that of Theorem 7.

If T is not known in advance, we can adapt algorithm Classify by applying a recent technique from [5] to obtain an algorithm with slightly worse competitiveness.

Theorem 8. *There exists a randomized online* SMP *algorithm with competitive ratio at most* $O\left(\prod_{i=1}^{\log^* T} \log^{(i)} T\right)$ *against oblivious adversaries that does not require knowledge of* T *in advance.*

Note that function $\log^{(i)}$ is defined as $\log^{(i)} T = \log\left(\log^{(i-1)} T\right)$ for $i > 1$ and $\log^{(1)} T = \log T$ while $\log^* T$ denotes the number of times we have to apply log to T in order to get a result smaller than 2.

References

1. Awerbuch, B., Azar, Y., Epstein, A.: The price of routing unsplittable flow. In: Proc. of the 37th Annual ACM Symposium on Theory of Computing (STOC '05), pp. 57–66 (2005)
2. Awerbuch, B., Azar, Y., Fiat, A., Leonardi, S., Rosen, A.: Online competitive algorithms for call admission in optical networks. Algorithmica 31(1), 29–43 (2001)
3. Awerbuch, B., Bartal, Y., Fiat, A., Rosen, A.: Competitive nonpreemptive call control. In: Proc. of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '94), pp. 312–320 (1994)
4. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press, Cambridge (1998)
5. Caragiannis, I., Fishkin, A., Kaklamanis, C., Papaioannou, E.: Randomized online algorithms and lower bounds for computing large independent sets in disk graphs. Discrete Applied Mathematics 155(2), 119–136 (2007)

6. Caragiannis, I., Flammini, M., Kaklamanis, C., Kanellopoulos, P., Moscardelli, L.: Tight bounds for selfish and greedy load balancing. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4051, pp. 311–322. Springer, Heidelberg (2006)
7. Christodoulou, G., Koutsoupias, E.: The price of anarchy of finite congestion games. In: Proc. of the 37th Annual ACM Symposium on Theory of Computing (STOC '05), pp. 67–73 (2005)
8. Czumaj, A., Vöcking, B.: Tight bounds for worst-case equilibria. In: Proc. of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '02), pp. 413–420 (2002)
9. Dobzinski, S., Schapira, M.: An improved approximation algorithm for combinatorial auctions with submodular bidders. In: Proc. of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '06), pp. 1064–1073 (2006)
10. Feige, U.: On maximizing welfare when utility functions are subadditive. In: Proc. of the 38th Annual ACM Symposium on Theory of Computing (STOC '06), pp. 41–50 (2006)
11. Feige, U., Vondrak, J.: The allocation problem with submodular utility functions. In: Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '06) (to appear)
12. Feldmann, R., Gairing, M., Lücking, T., Monien, B., Rode, M.: Nashification and the coordination ratio for a selfish routing game. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 514–526. Springer, Heidelberg (2003)
13. Fotakis, D., Kontogiannis, S., Koutsoupias, E., Mavronicolas, M., Spirakis, P.: The structure and complexity of Nash equilibria for a selfish routing game. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 123–134. Springer, Heidelberg (2002)
14. Hazan, E., Safra, S., Schwartz, O.: On the complexity of approximating k-dimensional matching. In: Arora, S., Jansen, K., Rolim, J.D.P., Sahai, A. (eds.) RANDOM 2003 and APPROX 2003. (Extended version as ECCC Report TR03-020) LNCS, vol. 2764, pp. 83–97. Springer, Heidelberg (2003)
15. Koutsoupias, E., Mavronicolas, M., Spirakis, P.: Approximate equilibria and ball fusion. Theory of Computing Systems 36(6), 683–693 (2003)
16. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: Meinel, C., Tison, S. (eds.) STACS 99. LNCS, vol. 1563, pp. 404–413. Springer, Heidelberg (1999)
17. Leung, J.Y-T. (ed.): Handbook of scheduling: algorithms, models, and performance analysis. CRC Press, Boca Raton (2004)
18. Mavronicolas, M., Spirakis, P.: The price of selfish routing. In: Proc. of the 33rd Annual ACM Symposium on Theory of Computing (STOC '01), pp. 510–519 (2001)
19. Motwani, R., Raghavan, B.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
20. Papadimitriou, C.: Algorithms, Games and the Internet. In: Proc. of the 33rd Annual ACM Symposium on Theory of Computing (STOC '01), pp. 749–753 (2001)
21. Roughgarden, T., Tardos, E.: How bad is selfish routing? Journal of the ACM 49(2), 236–259 (2002)
22. Yannakakis, M., Gavril, F.: Edge dominating sets in graphs. SIAM Journal on Applied Mathematics 38(3), 364–372 (1980)