The Roommates Problem

Online Matching on General Graphs

Ethan Bernstein[1] & Sridhar Rajagopalan[2]

CS Dept., UC-Berkeley

1 Introduction

The study of algorithms for on-line problems is a topic of current interest in algorithmic theory. In this paradigm, information (the input) is provided to the algorithm as a time series. The response (the output) of the algorithm is also viewed as a time series. The defining property of an on-line algorithm is that the response has to be *causal* with respect to the input. Thus, the response at time t can depend only on the inputs before and including time t. In contrast, one usually studies the "off-line" situation where the entire input is available to the algorithm before it computes any part of the output.

Sleator and Tarjan [5], suggested a measure of the efficacy of on-line algorithms. The measure, the *worst-case performance ratio*, compares the performance of an on-line algorithm to the performance of the optimal off-line solution. On one hand, this measure allows us to directly compare the off-line and on-line situations. On the other, it also provides us with a basis to compare two on-line algorithms.

In this paper, we study a variant of the on-line maximum matching problem. Such

[1]Supported by an NSF Graduate Fellowship.

[2]Supported by grants CCR-8896202 and IRI-8902813 from the NSF.

problems arguably have applications in issues such as job scheduling and resource allocations [4, 3]. We choose to state our problem here in the context of one such application, the *roommates problem.*

In this problem, the world consists of $2n$ people. The people arrive one at a time to an inn. Each informs the innkeeper which of the others he would agree to share a room with. The innkeeper must then assign the guest a room. We assume that the compatibility function is symmetric[3]. The performance of the innkeeper is measured by the number of rooms that are doubly occupied. In the weighted version of the problem, the compatibility function maps guest pairs to non-negative weights. The goal of the innkeeper is then to maximize the weight contained in doubly occupied rooms.

In this paper, we shall show that there is a deterministic algorithm that achieves a worst case performance ratio of $\frac{2}{3}$ in the unweighted case. We will also show that this is the best possible ratio for any deterministic algorithm. For the weighted problem, we will exhibit a deterministic algorithm that achieves a ratio of $\frac{1}{4}$ in the weighted case. In this case however, the bound is not tight. The best upper bound we have for deterministic algorithms is $\frac{1}{3}$.

2 Past work

The on-line matching problem in bipartite graphs was first considered in the paper of Karp, Vazirani and Vazirani [4]. In this problem, there are n boys and an equal number of girls. The girls arrive one at a time and indicate to a judge which of the boys she is interested in. The judge then has to choose one from among those she has expressed an interest in to

[3]Thus, if Alice would share a room with Bob, then Bob would not mind.

marry her off to. The aim of the judge is to maximize the number of married couples.

The main result in this paper is a randomized algorithm that achieves a performance ratio of $1 - \frac{1}{e}$ against oblivious adversaries. This number is also shown to be optimal.

Kalyanasundaram and Pruhs [3] consider the more general problem where the underlying graph, though still bipartite, is allowed to be weighted. However, they require that the bipartite graph be complete and that the weights be positive and satisfy the triangle inequality. They provide an algorithm that achieves an optimal performance ratio of $\frac{1}{3}$ on such graphs.

Notice that these previous works provide analysis only for bipartite graphs with restricted weightings. In contrast, our variant of online maximum matching is the first to allow non-trivial performance guarantees on general graphs with arbitrary weightings.

Finally, both [3] and the paper of Khuller, Mitchell and Vazirani [2] provide optimal competitive algorithms for variants of the minimum weighted matching problem.

3 The Problem.

Given an undirected graph $\mathcal{G}(V, E)$ with vertex set V and edge set $E \subseteq \begin{pmatrix} V \\ 2 \end{pmatrix}$, a *matching* M on $\mathcal{G}$ is a subset of E such that no two members of M share a common vertex. An instance of the *on-line matching problem*, consists of $\mathcal{G}$ plus some ordering σ on V. When an algorithm is run on $(\mathcal{G}, \sigma)$ it runs for $|V|$ phases. During the t^{th} phase the algorithm is given a list of the edges in E adjacent to $v_{\sigma(t)}$. At that time it may select one of those edges to add to its matching.

Sleator and Tarjan [ST] suggest the worst case competitive ratio as a measure of the efficacy of an on line algorithm. In situations where the goal is to *maximize* some quantity, the corresponding measure is the worst case performance ratio. We define it here in the context of maximum matchings. Let $\mathcal{M}$ be a maximum weight (cardinality) matching in $\mathcal{G}(V, E)$, and let Ψ be the matching picked by the (deterministic) on-line algorithm A. Then we define the performance ratio of A on $(\mathcal{G}, \sigma)$

$$P_{\mathcal{G},\sigma}(A) = \frac{|\Psi|}{|\mathcal{M}|}$$

And the worst case performance ratio of A is just

$$P(A) = \min_{\mathcal{G},\sigma} P_{\mathcal{G},\sigma}(A)$$

4 The weighted case.

We discuss here weighted on-line maximum matching. The better results that can be obtained in the simpler unweighted case are discussed in a subsequent section.

4.1 Conventions.

We will refer to the vertex that just arrived as v. If a vertex r arrives earlier than s we will say $r < s$. The set of vertices that have not yet arrived plus v form the set Y. We will use y to denote any generic element of Y. All vertices that we have not yet matched but have arrived in the past comprise the set X. We denote a generic element of X by x. The graph $(X, Y) = B$ is the bipartite graph on the vertices X and Y with weights as have been revealed to us. We will let $m(B)$ denote the size of the maximum matching on B and we will use u to denote the vertex that v is matched to (if one exists), in some such matching. At a generic moment of execution, any algorithm has two options.

- It could match v to some vertex $x \in X$. (MATCH occurs)
- It could add v to X. (NOMATCH occurs).

Every algorithm can be specified by defining the criterion that decides which option the algorithm will choose.

4.2 Two potential functions.

We define a potential $\beta(y)$ associated with each vertex $y \in Y$ that measures how important that vertex is to us. We define $\beta(y)$ for every $y \in Y$ as follows.

$$\beta(y) \stackrel{\text{def}}{=} m(B) - m(B - \{y\})$$

We also define a global potential function that measures the goodness of the current situation.

$$\Phi \stackrel{\text{def}}{=} m(B) + 2 \cdot \text{wt}\{ \text{ edges that we have matched so far } \}$$

The definition of Φ is exactly the maxim, "A bird in the hand is worth two in the bush." Notice that as in most potential based arguments we have a local quantity and a global potential. The argument consists of delineating the interplay between these two quantities.

We will use the superscript $+$ as in $\beta^+(\cdot)$ to denote the values of quantities after v has been processed. We will use Δf to denote the change of the quantity f with respect to time. Therefore, for any quantity f,

$$\Delta f \stackrel{\text{def}}{=} f^+ - f$$

4.3 The Algorithm.

Recall that u is the vertex that v is matched to in $\mathcal{M}(B)$. Our algorithm examines only two options.

- Match v to u. (MATCH happens)
- Add v to X. (NOMATCH happens)

Of these two options, it picks the one that maximizes Φ^+.

4.4 Analysis.

We use the following two lemmas to analyze the algorithm.

Lemma [Stability Lemma]: *The values of $\beta(\cdot)$ never decrease. Hence, $\forall y \in Y$*

$$\Delta\beta(y) \geq 0$$

Lemma [Payoff Lemma]: *The following lower bounds hold*

(a) $\Delta\Phi \geq \beta(v)$

(b) $\Delta\Phi \geq \frac{1}{2}(\mathrm{wt}(v, y) - \beta^+(y))$ *for every* $y \in Y$.

Theorem : *The algorithm that we have described achieves a worst case performance ratio of* $\frac{1}{4}$.

Proof : Consider any edge (r, s) in the graph. We will show that the sum of the gains in the global potential Φ when r and s arrive must be at least $\frac{1}{2}\mathrm{wt}(r, s)$. Therefore given any matching M in the graph,

$$\Phi_{\text{final}} \geq \sum_{(r,s)\in M} \frac{1}{2}\mathrm{wt}(r, s) = \frac{1}{2}|M|$$

Since the final potential is equal to twice the weight of the algorithm's matching, the algorithm achieves a performance ratio of at least $\frac{1}{4}$.

To see our claim about any edge (r, s), $r < s$, notice that the Payoff Lemma says that

$$\Delta\Phi|_{v=r} \geq \frac{1}{2}\left(\mathrm{wt}(r, s) - \beta^{+}(s)|_{v=r}\right)$$

The Stability Lemma tells us that

$$\beta(s)|_{v=s} \geq \beta^{+}(s)|_{v=r}$$

And the Payoff Lemma gives

$$\Delta\Phi|_{v=s} \geq \beta(s)|_{v=s}$$

Therefore, adding the above three inequalities we get,

$$\Delta\Phi|_{v=r} + \Delta\Phi|_{v=s} \geq \frac{1}{2}\mathrm{wt}(r, s)$$

□

We now prove our two lemmas. First we must introduce some new notation. We will denote by $B \leftarrow y$ the graph on $(X, Y + \{y\})$ and by $B \rightarrow y$ the graph on $(X, Y - \{y\})$. We will similarly define $x \rightarrow B$ and $x \leftarrow B$ as the graphs on $(X + \{x\}, Y)$ and $(X - \{x\}, Y)$ respectively. Hence the graph that results when MATCH happens is the graph $x \leftarrow B \rightarrow v$. Similarly, the graph that results when NOMATCH happens is $v \rightarrow B \rightarrow v$.

We begin by pointing out a few simple facts about matchings. Firstly, $m(B \leftarrow y) \geq m(B) \geq m(B \rightarrow y')$ for arbitrary vertices y, y'. Similarly, $m(x \rightarrow B) \geq m(B) \geq m(x \leftarrow B)$. We begin with the following technical lemma.

Lemma 1: $m(B) - m(B \rightarrow y) \leq m(B \rightarrow v) - m(B \rightarrow v, y)$.

Proof : This lemma says that $\beta(y)$ does not decrease if we remove a vertex (in this case v) from Y. We rearrange the terms and add $2m(B)$ from both sides so that it suffices if we show that

$$(m(B) - m(B \to v)) + (m(B) - m(B \to y)) \leq m(B) - m(B \to v, y)$$

Since v and y are on the same side of a bipartite graph, the right hand side of above inequality must consist of two disjoint alternating paths. The weights of these two paths represent upper bounds on the values of the two terms on the left hand side of the above inequality. □

Corollary 1.1: $m(B) - m(B \to y) \leq m(v \to B) - m(v \to B \to y)$.

Proof sketch: We can view the process of adding a vertex x to X as a process of deleting one from Y. We augment Y by the vertex y_∞ and X by the vertex v. The weight of the edge (v, y_∞) is ∞. The effect of deletion of the vertex y_∞ is the same as that of the addition of v to X. □

Lemma [Stability Lemma]: *The values of $\beta(\cdot)$ never decrease. Hence, $\forall y \in Y$*

$$\Delta\beta(y) \geq 0$$

Proof : If MATCH occurs then B^+ is a subgraph of B. Also, the maximum matching in B^+ is a subset of the matching in B. Therefore, any augmenting path in B^+ is there in B. Since, $\beta(y)$ essentially represents an augmenting path, we are done.

If NOMATCH occurs then the new B can be obtained by deleting v from Y and adding v to X. Lemma 1 and Corollary 1.1 tells us that $\beta(y)$ does not decrease in either case. □

Another way of viewing the quantity $\beta(\cdot)$ is that it is an investment that we have made. We will want to cash this investment sometime in the future. The lemma above tells us that these investments will not go sour. However, we also need two more guarantees. The first to tell us that we can reclaim our investments whenever we wish to. The second that these investments are sufficient for our objectives. We make these claims in the following lemma.

Lemma [Payoff Lemma]: *The following lower bounds hold*

(a) $\Delta\Phi \geq \beta(v)$

(b) $\Delta\Phi \geq \frac{1}{2}(\mathrm{wt}(v,y) - \beta^{+}(y))$ *for any* y *in* Y*.*

Proof : Since one option that we have is to match v to u, we are guaranteed that

$$\Delta\Phi \geq m(B) - m(u \leftarrow B \rightarrow v) \tag{1}$$

Alternately, since we could choose not to match v temporarily, we are guaranteed that for any $y \in Y$,

$$\Delta\Phi \geq \mathrm{wt}(v,y) - m(B) + m(B \rightarrow v, y) \tag{2}$$

The inequality (1) already gives us (a) of the lemma since $m(u \leftarrow B \rightarrow v) \leq m(B \rightarrow v)$. Adding the inequalities (1) and (2) gives us

$$2 \cdot \Delta\Phi \geq \mathrm{wt}(v,y) + m(B \rightarrow v, y) - m(u \leftarrow B \rightarrow v)$$

Since $m(u \leftarrow B \rightarrow v, y) \leq m(B \rightarrow v, y)$ we get

$$2 \cdot \Delta\Phi \geq \mathrm{wt}(v,y) + m(u \leftarrow B \rightarrow v, y) - m(u \leftarrow B \rightarrow v) \tag{3}$$

The inequality (3) is exactly the claim (b) in the event that MATCH happens. In the event that NOMATCH happens, Corollary 1.1 can be applied to (3) to get the claim. □

4.5 An upper bound on the performance ratio for the weighted case

Theorem 1: *Any on-line weighted matching algorithm has a worst-case performance ratio* $\leq \frac{1}{3}$.

Proof : Let A be some on-line weighted matching algorithm. We will show the existence of either a graph on which A achieves a performance ratio of $< \frac{1}{3}$, or a sequence of graphs on which the performance ratio of A approaches $\frac{1}{3}$.

Fix some $R > 3$. Given an $\epsilon > 0$, let $N = \frac{R-1}{\epsilon}$. We define $\mathcal{G}(\epsilon)$ as follows ...

- The vertex set of $\mathcal{G}(\epsilon)$, ordered by arrival, is $\{a_{-1}, b_{-1}, a_0, b_0, a_1, b_1, \cdots, a_N, b_N\}$.
- We define the edge set of $\mathcal{G}(\epsilon)$ recursively ...

  1. $\mathrm{wt}(a_{-1}, b_{-1}) = 1$,
  2. $\forall i > 0,\ \mathrm{wt}(a_i, b_i) = (R - i\epsilon) \cdot \mathrm{wt}(a_{i-1}, b_{i-1})$.
  3. $\forall i > 0,\ \mathrm{wt}(a_{i-1}, a_i) = \mathrm{wt}(a_i, b_i)$.

Consider running A on $\mathcal{G}(\epsilon)$. There must be some first edge which A chooses. Now, reveal the graph $\mathcal{G}'(\epsilon)$ to A, where each edge has the same weight as in $\mathcal{G}(\epsilon)$ until A chooses its first edge, as which point all unrevealed edges are switched to weight 0. Clearly the matching A chooses on $\mathcal{G}'(\epsilon)$ must be this single edge. Let r_ϵ be the ratio of the final non-zero edge revealed to the edge A chooses. It is easy to see that, for sufficiently small ϵ if r_ϵ is close to 1 (say < 1.1), then A achieves a performance ratio $< \frac{1}{3}$. So we need only consider the case where r_ϵ is bounded away from 1. For any $M < \frac{R-r_\epsilon}{\epsilon}$ the performance ratio A achieves on

$\mathcal{G}'(\epsilon)$ is at most

$$\frac{1}{r_\epsilon + \sum_{1\le i\le M} \prod_{1\le j\le i} \frac{1}{r_\epsilon + j\epsilon}}$$

If we fix M, then as $\epsilon \to 0$, the performance ratio of A on $\mathcal{G}'(\epsilon)$ approaches

$$\frac{1}{r_\epsilon + \sum_{1\le i\le M} \frac{1}{r_\epsilon^i}} = \frac{r_\epsilon - 1}{r_\epsilon^2 - r_\epsilon + 1 - \frac{1}{r_\epsilon^M}}$$

But since we could choose M to be arbitrarily large, and r_ϵ is bounded away from 1, the performance of A on $\mathcal{G}'(\epsilon)$ as $\epsilon \to 0$, approaches $\frac{r_\epsilon - 1}{r_\epsilon^2 - r_\epsilon + 1}$ This function has a maximum of $\frac{1}{3}$ at $r_\epsilon = 2$, so the best worst-case performance ratio A could possibly achieve as $\epsilon \to 0$ is $\frac{1}{3}$.

□

5 The Unweighted case.

We had said earlier that in the case that the graph $\mathcal{G}(V, E)$ was unweighted, stronger results could be obtained. In this section we show a tight bound of $\frac{2}{3}$ on the best possible performance ratio of deterministic algorithms for the unweighted problem.

5.1 Unweighted Matching Algorithm

Let $\mathcal{G}(V, E)$ be an arbitrary unweighted graph. Let v and Y be defined as in the weighted case. However, unlike the weighted case, the set X does not consist of all the vertices that have arrived but are not matched. When v arrives, the algorithm will either match v (MATCH), leave v unmatched and place it in X (NOMATCH), or discard v (DISCARD), in which case it will never attempt to match v. As before, $B = (X, Y)$ is the restriction of $\mathcal{G}$ to edges between X and Y. We now describe the algorithm.

1. Compute T, the new B that would result if NOMATCH were chosen.
2. If $m(B) \geq m(T)$, then MATCH v and u (if u exists), and DISCARD v otherwise.
3. Otherwise, NOMATCH : $B \leftarrow T$.

Lemma 1: *If $\exists x \in X$, adjacent to v, then* DISCARD *does not occur.*

Proof : Assume v has an edge to $x \in X$. Given any maximum matching M on B, removing the edge matching x and adding $\{x, v\}$ gives a maximum matching on B that matches v. Therefore, if $m(B) \geq m(T)$, then MATCH will occur rather than DISCARD . □

Lemma 2: *Let S be the set of maximum matchings on B. Let Ψ be the set of edges that our algorithm finally matches. Then $\exists M \in S$, such that $M \subseteq \Psi$*

Proof : By induction on $|Y|$. Notice that the induction reverses the progress of time!

Basis: $|Y| = 0$, Immediate.

Induction: Let M^+ be the maximum matching on B^+ such that $M^+ \subseteq \Psi$, which is guaranteed by the inductive hypothesis. We look at what the algorithm does with v:

If DISCARD , then $B^+ \subseteq B$, and M^+ itself is a maximum matching on B which is a subset of Ψ.

If MATCH , and v gets paired with u, then $M = M^+ \cup \{u, v\}$ is a maximum matching on B with $M \subseteq \Psi$.

If NOMATCH , then M^+ without its edge matching v is a maximum matching on B which is contained in Ψ. □

Corollary 2.1: *Any vertex put into X gets paired eventually.*

Proof : If $m(T) > m(B)$, then every matching on $B^+(=T)$ must match v. So if v is placed in X in step 3, then by Lemma 2, v must be paired eventually. □

A little case analysis can now be used to give the following theorem.

Theorem 1: *Let $\mathcal{M}$ be a maximum matching on $\mathcal{G}$. Every augmenting path p in $\mathcal{M} \oplus \Psi$ has at least 5 edges in it. Hence, our algorithm achieves a performance ratio of $\frac{2}{3}$.*

Proof : In the following argument we will use X^a and B^a to represent the states of X and B when the vertex v_a arrives. We show every augmenting path p in $M \oplus \Psi$ is such that $|p| \neq 1$ and $|p| \neq 3$. We look at the easy case first.

Assume there is an augmenting path $p = (v_a, v_b)$. Let wlog $v_a < v_b$. Then, since v_a is unpaired, v_a is immediately discarded (Corollary 2.1). But, v_a could only be discarded if v_b is matched in every maximum matching on B^a. Therefore, v_b is eventually paired with some $x \in X^a$ (Lemma 2).

The other case is slightly more complicated. Let, if possible, there be an augmenting path $p = (v_a, v_b, v_c, v_d)$. We prove a contradiction by examining the relative arrival order of these four vertices. Noting symmetry, we need to consider only the two cases in which v_a is the first vertex to arrive or v_b is the first vertex to arrive. We consider these one at a time.

If v_a were to arrive first then the same argument as above tells us v_b gets paired to some vertex in X^a. However, since $v_c \notin X^a$, this contradicts the fact that v_b and v_c are paired.

Now assume, v_b is the first to arrive. If v_a arrives second, it will be paired (Lemma 1).

$\Rightarrow\Leftarrow$ If v_c arrives second, it must be immediately paired to v_b. But this implies that either v_a or v_d must be matched in every maximum matching on B^c. Therefore, the algorithm will eventually pair v_a or v_d (Lemma 2). $\Rightarrow\Leftarrow$ Finally, if v_d arrives second, it must be the case that v_a is matched in every maximum matching on B^d that matches v_b with v_c. But, then v_a must eventually be paired (Lemma 2). $\Rightarrow\Leftarrow$

This completes the proof that there is no one or three edge augmenting path in $\mathcal{M} \oplus \Psi$.

$\square$

5.2 An upper bound.

We will now show that $\frac{2}{3}$ is the best possible performance ratio for any deterministic algorithm. Consider the input instances $\langle P_5, \sigma_1 \rangle$ and $\langle P_5, \sigma_2 \rangle$ where P_5 is the simple path with 5 edges (and 6 vertices) and $\sigma_1 = (2, 3, 1, 4, 5, 6)$ and $\sigma_2 = (3, 4, 1, 2, 5, 6)$. Any deterministic algorithm will fail to achieve a matching of size 3 on one of these inputs. The reason is that the situation encountered by the algorithm after 2 vertices have arrived is exactly the same for both the examples. However, in one case, MATCH is a bad decision and in the other NOMATCH (or DISCARD) is. Since any deterministic algorithm must do one of the two, it fails on the corresponding input. $\square$

6 Conclusion

The following are the most interesting open problems.

1. Is it possible to tighten the gap in the weighted case? This paper shows that the best possible worst case performance ratio for deterministic algorithms is somewhere in the

interval $[\frac{1}{4}, \frac{1}{3}]$. We also note that the analysis of the algorithm detailed in Section 3 is tight (even though it does not seem to be). There is a class of graphs on which this algorithm achieves competitive ratios arbitrarily close to $\frac{1}{4}$. In this case, we conjecture that the algorithm that matches v to any $x \in X$ (and not just u) greedily on Φ achieves a better performance ratio. It is not true, however that this algorithm dominates our algorithm. There are cases where the matching it gets is smaller. However, in such cases the first algorithm does extremely well.

2. What can be said about randomized algorithms working against oblivious adversaries [1]? It is possible to show that any randomized algorithm can not perform better than $\frac{4}{5}$ in the unweighted case by using an extension of the analysis in this paper via Yao's lemma [6]. On the other hand there is a deterministic $\frac{2}{3}$ algorithm.

7 Acknowledgements.

We wish to acknowledge the contributions of Umesh Vazirani to this work.

References

[1] S.Ben-David, A.Borodin, R.Karp, G.Tardos, A.Wigderson. "On the Power of Randomization in On-Line Algorithms", *Proceedings of the 22nd ACM STOC*, pp. 379–386, 1990.

[2] S.Khuller, S.Mitchell, V.Vazirani. "On-Line Algorithms for Weighted Matching and Stable Marriages", Technical Report TR 90-1043, Department of Computer Science, Cornell University, 1990.

[3] B.Kalyanasundaram, K.Pruhs. "On-Line Weighted Matching", *Proceedings of the SODA,1991, San Francisco.*

[4] R.Karp, U.Vazirani, V.Vazirani. "An Optimal Algorithm for On-Line Bipartite Matching", *Proceedings of the 22nd ACM STOC*, pp. 352–358, 1990.

[5] D.Sleator, R.Tarjan. "Amortized Efficiency of List Update and Paging Rules", *Communications of the ACM*, vol. 28, pp. 202–208, 1985.

[6] A.Yao. "Probabilistic computations: Toward a unified measure of complexity." *Proceedings of the 18th IEEE FOCS*, pp. 222–227, 1977.