Chiang Mai J. Sci. 2011; 38 (Special) : 17-30
www.science.cmu.ac.th/journal-science/josci.html
Contributed Paper

Wellness, Social Networking, and Algorithms

Raymond Greenlaw
Computer Science Department, 572M Holloway Road, Stop 9F United States Naval Academy Annapolis, MD 21402-5002. and Elbrys Networks, Inc. 282 Corporate Drive Portsmouth, New Hampshire 03801.
e-mail: greenlaw@usna.edu

Received: 15 July 2010
Accepted: 31 January 2011

ABSTRACT

This paper is one of the first written that ties together the areas of wellness, social networking, and computational-complexity theory. Our goal is to integrate these three fields in order to develop a system in which we can help people improve their well being. There are over one billion *obese* people in the world and that number is growing alarmingly fast. We must find a way to motivate unhealthy people to take care of themselves, so as to put less strain on economies around the world, less strain on natural resources, less strain on the health-care system, less strain on nearby individuals, and, most importantly, to save lives and to reduce suffering. We have gone from hunters and gathers to sitters and twitterers. As our standard of living increases, so does our size. While eating high-caloric foods, many people watch TV or play with the Internet all day long. And one particular Internet application, social networking, is amazingly popular-nations around the world have a significant percentage of their populations engaged in social-networking sites such as Facebook, Flickr, Friendster, hi5, LinkedIn, MySpace, Orkut, Qzone, and so on. In this paper we leverage the popularity of social networking to improve people's wellness. We propose a WELLNESS PROFILE MODEL that captures a social-network members' characteristics, preferences, activities, vitals, and other relevant information. We formulate *dynamic-matching problems* whose focus is to find groups of individuals with similar interests and time constraints, so that they can participate in activities together thereby contributing to each other's well being. We examine the computational complexity of the problems from both the sequential and parallel perspectives. Our results tell us how hard these problems are to solve. That is, we determine the complexity of these problems. The problems defined here involve a new type of dynamic matching, as opposed to traditional matching as is done in many of the current Internet social-networking sites and traditional graph theory. We intend to implement restricted versions of some of the algorithms in the systems being developed by Elbrys Networks, Inc.

Keywords: algorithms, computational-complexity theory, dynamic matching, social networking, mobile wellness applications, wellness profile model.

1. INTRODUCTION

The world has seen an explosion in the size of the human being [1]. In the 1970s the seat-belt extender was unheard of on airplanes (barring sumos), but now it is common to

have individuals requiring extenders to buckle up, and numerous airlines, including Air France and KLM, charge obese individuals roughly double the standard fare because such folks no longer fit in one seat. In the United States of America (USA), there is even a TV show called the *Biggest Loser* in which grossly overweight people try to lose enormous amounts of weight. And, obese individuals have far more health problems than (well) healthy individuals [2-5]. Skyrocketing health-care costs are putting tremendous strains on economies around the world. For example, in 2009 the USA spent 17% of the GDP on health care [6], and the growth rate of health-care spending is far outpacing the growth rate of the consumer price index. We must find ways to improve the health of the average person; we simply can not pay for the costs associated with a world population where roughly 20% of all individuals are obese. Medical data shows that 75-80% of poor health issues are preventable [7].

Internet social-networking sites have become extremely popular in recent years. A number of the best-known social-networking sites have several hundred- million members and are steadily increasing in size. We would like to take advantage of people's interest in social networking to improve their wellness. In one area of wellness, for example, it is known that obese individuals have far more health problems than healthy individuals [2-5]. In this paper we develop a WELLNESS PROFILE MODEL designed to capture characteristics, preferences, activities, vitals, and other relevant information of its members. We formulate *dynamic-matching problems* to find groups of individuals with similar interests and time constraints, so that they can participate in wellness activities together and motivate each other to continue to improve their wellness levels. Our model is a step toward producing an environment in which groups of individuals can be matched in real time. The problems defined here, the algorithms developed, and the computational-complexity results proved involve a new type of dynamic matching, as opposed to traditional matching, as is done in many of the Internet social-networking sites. We intend to incorporate a specific implementation of the model into the systems of Elbrys Networks, Inc. [8] so that individuals can arrange wellness activities with others located all over the world. We plan to incorporate special cases of our matching algorithms into the systems too. In future work we also plan to incorporate a rewards system into the model to provide further incentives for individuals to improve their wellness.

Here we provide a brief review of some of the previous work on graph matching; we refer the reader to [9] for basic concepts. We also give some background on the field of computational-complexity theory, and in Appendix A we include more details for the non-specialist. Complexity theory is a subfield of computer science used to classify the computational difficulty of solving problems. Problems having extremely high complexity may not be feasible to solve on the fastest computers available now nor at any point in the future. Perhaps the most well known of the computationally intractable problems is the TRAVELING SALESMAN PROBLEM [10]. In this problem the input consists of a set of cities, the distance between each pair of cities, and a bound b. The bound b is the maximum distance that we would like the salesman to have to travel. The problem is to determine if there exists a traveling salesman's tour of cost less than b, where each city gets visited. All the known solutions to this problem require an exponential amount of time in the number of cities. So, for example, an instance of the problem with just sixty cities would require on the order of 2^{60} (roughly,

10^{18}) computational operations. And, even if executing one billion instructions per second, the problem would require about a full year to solve. The TRAVELING SALESMAN PROBLEM is an *NP-complete* problem.

Some of the matching problems that have been studied previously are as follows:

- MINIMAL MAXIMAL MATCHING PROBLEM
- MAXIMUM SUBGRAPH MATCHING PROBLEM
- MULTIPLE CHOICE MATCHING PROBLEM
- 3-DIMENSIONAL MATCHING PROBLEM
- NUMERICAL 3-DIMENSIONAL MATCHING PROBLEM
- NUMERICAL MATCHING WITH TARGET SUMS PROBLEM
- ALTERNATING MAXIMUM WEIGHTED MATCHING PROBLEM

All of these problems are known to be NP-complete, as shown in [10-14], respectively. Thus, many variants of matching are known to be intractable. These problems all have a computational complexity that is equivalent to that of the TRAVELING SALESMAN PROBLEM. So, there are no practical algorithms for solving these problems. As the problem size grows, it would take an enormous amount of computation time to solve the any one of these problems for just one instance. The references give more details about many variants of these problems. The interested reader may want to see some of the following references for more information: [15-28].

The remainder of this paper is as follows: section 2 presents the specification of the WELLNESS PROFILE MODEL; section 3 provides a specific instance of the model for the system; section 4 discusses the matching problems that we consider; section 5 classifies the complexities of our matching problems; section 6 contains a description of a restricted practical dynamic-matching algorithm for implementing the WELLNESS PROFILE MODEL in the system; and section 7 presents conclusion and open problems. Appendix A gives background on computational-complexity theory.

2. WELLNESS PROFILE MODEL

The goal of this section is to describe the WELLNESS PROFILE MODEL. Before presenting the formal details of the model, we describe the intuition behind the model. The idea is to build groups of members dynamically according to their characteristics, preferences, number of desired activity partners, availability, and vital statistics. Imagine a social network with one-hundred-million members each of whom have specified such information. Imagine they have technology on their wrists or clipped to their belts that is capable of linking them together with video, voice, and vital statistics; as in the system of Elbrys Networks, Inc. Then, for example, Ray in Chiang Mai, Thailand could have a run in real time with the following runners: Paul in Eliot, Maine and Robert in Durban, South Africa; all of whom run the same pace, want to run 10K, and have the same availability. Individuals in a group motivate each other to run faster and further, and running together is much more fun. Note that in contrast, to say, dating sites, here, when the tuple (Ray, Paul, Robert) is formed, these individuals can not be matched to run with others. They are *dynamically* "taken out" of the pool of available runners. Note too that "run" can be replaced with any type of wellness activity.

Armed with this intuition, we now present the formal definition of the model. The model is designed to capture any type of wellness activity and framework. Let $\mathbf{N}$ =

$\{1, 2, 3, \ldots\}$ be the set of natural numbers.

Definition 2.1 WELLNESS PROFILE MODEL

A WELLNESS PROFILE MODEL (WPM) *is an 8-tuple* (M, C, P, I, A, T, S, V), *where each component is specified in the following:*

1. *The set of members is given by* $M = \{m_1, \ldots, m_k\}$, *where* $k \in \mathbf{N}$.
2. *The set of characteristics is specified by* $C = \{c_1, \ldots, c_l\}$, *where* $l \in \mathbf{N}$. *Associated with each* c_i, $1 \leq i \leq l$, *is a set* $\Gamma(c_i)$ *that represents the set of all possible values for each* c_i. *There is a linear order* t_i *on* $\Gamma(c_i)$ *for* $1 \leq i \leq l$.
3. *The set of preferences is given by* $P = \{P_1, \ldots, P_k\}$, *for* $1 \leq i \leq k$. *For each* i *in this range* $P_i = (v_{i1}, \ldots, v_{il})$, *where vij with* $1 \leq j \leq l$ *is the value in* $\Gamma(c_j)$ *for member* i *on characteristic* j.
4. *The intervals representing the number of desired partners are specified by* $I = \{[n_{ia}, n_{ib}] \mid 1 \leq i \leq k$ *and all* $n_{ia}, n_{ib} \in \mathbf{N}$ *with* $n_{ia} \leq n_{ib}\}$. *Note that we do count a member himself in this specification.*
5. *The set of activities is given by* $A = \{a_1, \ldots, a_o\}$ with $o \in \mathbf{N}$. *For* $1 \leq i \leq k$, *the set of desired activities for member* i *is given by* $Ai \subseteq A$.
6. *For* $1 \leq i \leq k$ and $r \in \mathbf{N}$, $T = \{T_1, \ldots, T_k\}$ *is the set of available times, where* $T_i = \{(t_{i,s1}, t_{i,e1}), \ldots, (t_{i,sr}, t_{i,er})\}$ *with* $sp, ep \in \mathbf{N}$ *and* $sp < ep$ *for* $1 \leq p \leq r$. *We call* $[t_{i,sx}, t_{i,ex}]$ *the xth-interval of availability for member* i. *The duration of such an interval is* $t_{i,ex} - t_{i,sx}$. *Without loss of generality, we assume each member provides the same number of intervals.*
7. *The set of vital statistics is given by* $S = \{\alpha_1, \ldots, \alpha_q\}$, *where* $q \in \mathbf{N}$. *Associated with each* α_i, $1 \leq i \leq q$, *is a set* S_i *that represents the set of all possible values for* α_i.
8. *The members' vital statistics are given by* $V = \{V_1, \ldots, V_k\}$, *where for* $1 \leq i \leq k$, $V_i = (\beta_{i1}, \ldots, \beta_{iq})$ and v_{ij} with $1 \leq j \leq q$ *is the value in* S_j *for member* i *on vital statistic* j.

We have defined the model very generally so that it can capture the profiles and related information of nearly any situation involving the wellness properties and fields of any group of users. One can think of the model as a database of information about a group of users.

Now we want to group like users together. And, the next definition that we provide captures this notion formally. We say that the *constraints of the model are met* by a group of members $G = \{m_1, \ldots, m_c\}$ for $c \in \mathbf{N}$ if the preferences of the members are the same; c is in the interval of the number of desired partners for each member in G; there is some activity a common to all members; there is some time period t common to all members; and for all members in G the *vital statistics of each pair of members are within range.* Note that, in general, we do not require an exact match on the vital statistics but just that they fall *reasonably close* to each other. Here reasonably close will depend on the particular instance of the model and the particular vital statistic. In future work we plan to examine this type of fuzzy multi-dimensional matching further. We say that members in the same group are *matched.*

We plan to extend the model to include a rewards system as well. That system will be based on social interactions, as well as duration and intensity of activity, and be general enough to incorporate other types of evaluation, for example, group weight loss or reduction in the summation of the group members' resting heart rates. To make things more concrete, in the next section we present a small example of the model, as it will be used in the system.

3. AN INSTANCE OF THE WELLNESS PROFILE MODEL FOR THE SYSTEM

In general, we anticipate our model being implemented in social networks having hundreds of millions if not a billion members. Here we offer an example of how the system

might utilize dynamic matching in order to provide a user some suggestions of workout partners for the day. Keep in mind that the system and model are capable of handling a wide range of wellness activities. The idea is that the group of users (perhaps remote from each other) would exercise at the same time and be able to compare their personal sensor readings. This example is by necessity simplified. Note that in our simplification we are not completely precise in the use of notation in this example.

1. Members are Anthony, Bruce, Carla, David, Edith, and Francine.
2. Characteristics are as follows:

r = reduce stress	e = find energy
f = have fun	p = push myself
a = be admired	s = spend time with others
c = compete against others	m = manage my illness
l = lose weight	i = improve fitness
o = be outdoors more	h = help environment
b = challenge my brain	j = improve my career
g = help others	t = spend more time with family

Also, the possible values for each characteristic in this case are the same and given by 0 = "not important" and 1 = "important." The linear order is given by $0 < 1$.

3. The preferences of our members in order are determined from their stated motivations and their selected challenge motivations as follows:

Member	Preferences
Anthony	(f, p, i)
Bruce	(r, b, t)
Carla	(f, p, i)
David	(p, c, o)
Edith	(r, e, l)
Francine	(p, o, g)

So, for example, Carla is interested in having fun, pushing herself, and improving fitness.

4. The intervals for the number of desired partners in order are as follows:
   [3, 5], [2, 3], [1, 3], [1, 1], [2, 5], and [2, 5].
   So, for example, Bruce would like two or three partners.
5. The set of activities is given by c = cycling, d = dance, r = running, s = swimming, and w = walking. The desired activities in order are given as follows:

Member	Desired Activities
Anthony	$\{c, s, w\}$
Bruce	$\{r, w\}$
Carla	$\{c, d, r, w\}$
David	$\{c, s\}$
Edith	$\{c, d, r, s, w\}$
Francine	$\{c, d, r, s, w\}$

So, for example, David wants to cycle or swim.

6. The available times, using the 24-hour clock, in order are given as follows:

Member	Available Times
Anthony	{(7, 8), (8, 9), (16, 17), (17, 18)}
Bruce	{(5 : 30, 6 : 30), (6 : 30, 8), (9, 10), (20, 21)}
Carla	{(7, 8), (9, 10), (10, 11), (16, 17)}
David	{(9, 10), (16, 17), (17, 18), (20, 21)}
Edith	{(9, 10), (15, 16), (16, 17), (18, 19)}
Francine	{(6 : 30, 8), (9, 10), (16, 17), (20, 21)}

7. The vital statistics are b = body mass index (bmi) and h = resting heart rate. The possible values for these statistics come from **N**.
8. The members' vital statistics are given as follows:
   {(21, 72), (25, 80), (20, 70), (21, 74), (35, 95), (19, 58)}

So, for example, Francine has a bmi of 19 and has a resting pulse of 58 beats per minute.

In this instance we see that Anthony and Carla have the same characteristics, both would be happy with three activity partners, and both want to cycle or walk at 7-8 am or 4-5 pm. If Anthony and Carla are matched, they will be taken out of the pool. Notice that although Edith is available for cycling at 4 pm, she is not matched with Anthony and Carla because Edith is not fit and is significantly overweight; her heart rate is also considerably higher.

4. DYNAMIC-MATCHING PROBLEMS

In this section we present several definitions regarding matching, and define several important matching problems related to our model. Throughout this section let $W=(M, C, P, I, A, T, S, V)$ be an instance of the WPM. Intuitively, a matching is the result of creating groups of people who have similar needs with the caveat that a person can only be a member of one group. We will explore several different types of matching. Keep in mind that our goal is to match as many users as possible, while meeting their requirements, and be able to find that matching as efficiently as possible. It turns out that computing matchings in this setting is hard. And, in this section we give some *absolute* results about the complexity of solving the problems posed. By this we mean that we will show that it is impossible for *anybody* to come up with efficient algorithms for some of these problems. We prove that such algorithms do *not even exist*. Naturally, to prove such a result does get a bit technical, but we include intuitive statements with all of our results.

Let us begin by formally defining what we mean by a matching in the Well- ness Profile Model.

Definition 4.1 MATCHING IN WPM

*A matching in a WPM W is a set of groups of members G_1 ,..., G_p for some $p \in$ **N** such that for each G_i, $1 \le i \le p$, the constraints of the model are met by G_i and for any two groups G_i and G_j, $1 \le i \ne j \le p$, G_i and G_j share no members.*

Of course, we would like to match as many people as possible. And, the next defintion captures the idea of having a large matching. This definition says that there are matchings which cannot be extended. That is, the matching puts as many people in groups as possible, and no more members can be added to any of those groups because they would not be joining a group of members having similar needs. Thus, we see that it is important how groups form and arise.

Definition 4.2 MAXIMAL MATCHING IN WPM

*A maximal matching in a WPM W is a matching G_1 ,..., G_p for some $p \in$ **N** such that there is no member, who is not already in a group, who can be added to any G_i and have the constraints of the model met, and further, no new group can be formed from members who are not already in some group and have the constraints of the model met. The members of a group G_i are called a matched group.*

Note that in the definition of maximal matching, the second condition states that even no singleton "group" can be formed, which is often the case since many people will not workout alone. Interestingly, there could be many many maximal matchings for a given WPM. Some of these maximal matchings might be very big, but others might not be so

big. We would like to capture the notion of the largest possible matching. The next definition does this. Essentially, it defines the matching which has the largest possible number of members matched. There is no other matching of greater size; there could be others of equal size though. That is, a maximum matching will be the maximal matching of largest size. It has the most possible total number of members in its groups.

Definition 4.3 MAXIMUM MATCHING IN WPM

The size of a matching is the total number of members in all the groups that make up the matching. A maximum matching in a WPM W is a maximal matching of maximum cardinality.

The last definition captures the notion of a perfect matching. It is perfect in the sense that everyone is happily matched. No one has to go it alone in this case. For a given set of constraints, it is often the case that no perfect matching exists, just as we find in everyday life, things are often not perfect.

Definition 4.4 PERFECT MATCHING IN WPM

A perfect matching in a WPM W is a maximum matching in W in which every member is in a matched group.

We plan to explore other interesting problems based on the model as well.

5. COMPLEXITY OF DYNAMIC-MATCHING PROBLEMS

Here we characterize the complexity of the problems defined in section 4. Perhaps the most natural algorithm for computing a maximal matching in a WPM is to process each member in the given input order and begin considering that member for addition to any of the previously formed groups in order or to start a new group using that member as the first entry, if that member did not meet the constraints of the model for any other group. We call the matching formed by this greedy algorithm the *lexicographically first maximal matching in* a WPM. Think of this matching as the one that would occur first in "alphabetical order," where the alphabetical order is taken over the numbering of the members. Such a matching may be a good approximation to a maximum matching, or it may be a very poor approximation. The quality of the approximation will depend on the ordering of the members, and their preferences. Of course, we are interested in matching as many individuals as possible.

In this section we pose three questions and provide answers to them, providing proofs of each answer's correctness. The questions posed and the answers proven are as follows:

1. (Theorems 5.4, 5.5, and 5.6)

Will a given user m be placed in a particular group in a matching? Here we have in mind using the lexicographically first maximal matching algorithm to find the matching. And, how difficult is this problem to solve? We prove that this question may be answered in $O(n^2)$ time (Theorem 5.4; meaning, roughly, that if the number of users input is n, then the time required to answer the question is some constant times n^2. Such an algorithm is considered practical, yet could still take a long time to run. So, for a problem with 1,000 users, we would need about 1,000,000 computational steps to solve the problem. Problems with polynomial running times, nk, for some constant k are considered feasibly solvable. And, of course, the smaller the k the better the solution. The second two theorems combine to prove that this particular problem is as difficult to solve as any other problem that can be solved in polynomial time. This result (Theorem 5.6) shows that

this problem is indeed a very difficult problem to solve.

2. (Theorem 5.7)

How hard is it to find a very large matching, that is, a maximum matching? We can answer this question by asking for larger and larger values of k if there is a matching where the number of groups in the matching is greater than or equal to k? We prove that this problem is not solvable in a reasonable amount of time for any large data sets (Theorem 5.7). This result shows that finding a maximum matching is much much harder than finding a maximal matching (up to some technical conditions).

3. (Corollary 5.8)

How hard is it to compute whether or not there is a perfect matching, that is, a matching where everyone is put in a group? This matching is one where no one is left out. (As we noted earlier, in some case such a matching does not even exist.) We prove that this problem is not solvable in a reasonable amount of time for any large data sets by showing that this problem is at least as hard as finding a maximum matching (Corollary 5.8). This result means that there are no practical algorithms that *will ever be discovered* for solving this problem (up to some technical conditions).

The following theorems are necessarily technical, but readers who are not interested in the technical details may skip to the start of section 7, as we already covered the essence of these results previously.

As is traditional in complexity theory, we define decision problems based on the definitions from section 4 and study the complexity of these problems (see [29] for further background on computational-complexity theory and decision problems). Throughout this section we use n to denote the size of an encoding of an instance $W = (M, C, P, I, A, T, S, V)$ of the WPM.

Definition 5.1 WPM MAXIMAL MATCHING PROBLEM (WMAXIMALM) GIVEN: *Let $W = (M, C, P, I, A, T, S, V)$ be an instance of the WPM and m a member of M.*
PROBLEM: *Is m in the lexicographically first maximal matching of W? That is, is m matched by the greedy algorithm described earlier?*

Note that W MAXIMAL M stands for Wellness Profile Model Maximal Matching Problem.

Definition 5.2 WPM MAXIMUM MATCHING PROBLEM (W MAXIMUM M) GIVEN: *Let $W = (M, C, P, I, A, T, S, V)$ be an instance of the WPM and $k \in \mathbb{N}$.*
PROBLEM: *Is there a matching of W of size greater than or equal to k?*

Definition 5.3 WPM PERFECT MATCHING PROBLEM (W PERFECT M) GIVEN: *Let $W = (M, C, P, I, A, T, S, V)$ be an instance of the WPM.*
PROBLEM: *Does W have a perfect matching?*

We now prove some theorems about the complexity of these problems.

Theorem 5.4 *The* WPM MAXIMAL MATCHING PROBLEM *can be solved in $O(n^2)$ time.*
Proof. The greedy algorithm described at the beginning of this section can be used to solve this problem. The algorithm needs to process each member once and determine which group, if any, that the member fits in.

In the parallel setting, we have the following result.

Theorem 5.5 *The* WPM MAXIMAL MATCHING PROBLEM *is P-hard under* NC many-one reducibility.
Proof. Given an instance of the standard

LEXICOGRAPHICALLY FIRST MAXIMAL MATCHING PROBLEM $G = (V, E)$ and e (see [13]), we can form an instance of the WPM W in NC as follows: for each node in V, we introduce a member in M in W. The members have the same order as the vertices. All members have the same characteristics and preferences, desire just one other activity partner, have the same single time availability, and have the same vital statistics. For each edge in E we have a "new" activity that is only shared by its member endpoints. So, each member is a potential match for his neighbors if he has not been matched yet. It is now clear that $e = \{u, v\}$ will be in the lexicographically first maximal matching of G if and only if members u and v are matched in W.

Theorem 5.5 shows that there is no poly-logarithmic time, polynomial number of processors algorithm for solving AMAXIMALM, unless NC = P.

Theorem 5.6 *The* WPM MAXIMAL MATCHING PROBLEM *is P-complete under NC many-one reducibility.*

Proof. The result follows from Theorems 5.4 and 5.5.

Theorem 5.7 *The* WPM MAXIMUM MATCHING PROBLEM *is NP-hard under polynomial-time many-one reducibility.*

Proof. As we noted in the introduction, the 3-DIMENSIONAL MATCHING PROBLEM is NP-complete [13]. An instance of this problem is a set $U \subseteq X \times Y \times Z$, where X, Y, and Z are disjoint sets having the name number of elements $q \in \mathbf{N}$. The problem is to determine if U contains a subset $U' \subseteq U$ such that $|U'| = q$ and no two elements of U' agree in any coordinate. From U we can be build an WPM $W = (M, C, P, I, A, T, S, V)$, where for each $x \in X$, $y \in Y$, and $z \in Z$, we introduce a member $m \in M$; in total $3q$ members. If the triple $(x, y, z) \in U$, then we set the specifications in the model so that the constraints of the model are met by this group with an activity unique to this group. We set one time interval, and it is the same for all members. The only other important specification is that each member desires three activity partners, including himself. It is not hard to see that there is a 3-D matching in U if and only if there is a maximum matching in W of size q. The reduction can be performed in polynomial time.

Corollary 5.8 *The* WPM PERFECT MATCHING PROBLEM *is NP-complete.*

Proof. This result follows via the reduction given in Theorem 5.7, and the fact that we can guess and check to see if a matching is perfect in polynomial time.

There is an algorithm for constructing a perfect matching in an undirected graph very fast in parallel using randomization [23], but Corollary 5.8 shows that no such fast parallel algorithm exists for the WPM PERFECT MATCHING PROBLEM, unless NC = NP. Note that W PERFECT M is in NP, but we have not shown W MAXIMUM M is in NP.

6. PRACTICAL DYNAMIC-MATCHING ALGORITHM

The results presented thus far indicate that some versions of the dynamic- matching problems are very difficult to solve. Here we focus on a specific instan- tiation of the WPM for the system. For the sake of this discussion, we assume the characteristics of the model are the sixteen characteristics of the system, as specified in the example in section 3. The values for these characteristics are binary, either 0 = "not important" or 1 = "important." We assume the same activities are available as in the example provided, and also assume the same two vital statistics. To simplify matters we assume each person desires a single activity partner. We further assume desired time slots

start on the hour and last for exactly one hour. Given these assumptions we have developed the following algorithm to form dynamic matches in the system.

```
Dynamic-Matching Algorithm
{ Input: Restricted WPM W = (M, C, P, I ,
A, T , S, V ). }
{ Output: A matching in W. }
begin
{ Comment: Place members in buckets. }
for each member place that member in one
  of 2^16 buckets according to the member's
  preferences;
{ Comment:
  Match as many members as possible. }
  for each time slot {
   for each nonempty bucket {
    for each activity {
     for each set of differing vital statistics {
      G ← graph consisting of all members
        who fall reasonably close;
        run a maximum matching algorithm
        on G;
        output all matched pairs in G;
      }
    }
   }
  }
end.
```

Note that this algorithm gives us a starting point for finding dynamic matches in the system. The algorithm is certainly not guaranteed to find an overall max- imum matching in a WPM, but it does find a type of maximal matching in a WPM. The maximal matching in a WPM found by the DYNAMIC-MATCHING ALGORITHM is a type of greedy matching. Notice that the quality of the matching found by the algorithm will depend on many factors. Also, note that users who specify more than one available time slot may be matched up multiple times per day. Although some users will want to participate in more than one activity per day, others may not, and the implementation used in the system will need to take this fact into account. The following theorem describes the running time of the algorithm, and it relates the running time to the running time of the best known algorithm for maximum matching.

Theorem 6.1 *Let n be the number of members in an instance of the* WELLNESS PROFILE MODEL, *and let* $\mathcal{M}$ *denote the running time of the best-known maximum matching algorithm. The* DYNAMIC MATCHING ALGORITHM *runs in* $O(n\mathcal{M})$.

Proof: Notice that the maximum matching algorithm is run at most n times on a graph having at most n vertices. Therefore, the bound stated in the theorem holds.

For deterministic algorithms the best - known maximum - matching algorithm runs in time $O(\sqrt{(|V|)}\ |E|)$, where V is the vertex set of the graph and E is the edge set [15, 16]. Thus, for an instance of the WELLNESS PROFILE MODEL having n members, the DYNAMIC MATCHING ALGORITHM runs in $O(n^{3.5})$ time. Note that the function $n^{3.5}$ grows quite quickly, but for several thousand users the algorithm should be reasonable. Also, "many" of the at most n iterations of the algorithm run are likely to be on fairly small graphs. However, in a system having millions of users the algorithm would not be practical. We expect the system to be deployed in local communities and high schools, and typically, these domains would consist of at most a few thousand members. We will need to test the algorithm on real data to see how well it works in practice.

7. CONCLUSIONS AND OPEN PROBLEMS

This paper is one of the first written to tie together wellness, social networking, and computational-complexity theory. We defined a WELLNESS PROFILE MODEL and the

concept of dynamic matching within the model. Then we defined a number of matching problems using the framework of the model. We showed how to find an approximate matching of members in the model, but unfortunately, finding that approximation led to a P-complete problem. This result means that there is very little chance of developing a fast parallel algorithm for this problem. We then showed that to find either a maximum or perfect matching in the model was NP-hard and NP-complete, respectively. Since a typical implementation of the model is expected to be drawn from a social network having hundreds of millions of users, one can not hope to have an efficient exact solution to the matching problem. This fact suggests that we need to develop good heuristic solutions for matching or examine special cases. We also developed a sequential algorithm for the system, and showed that the running time of the algorithm is $O(n\mathcal{M})$, where $\mathcal{M}$ denotes the best-known running time for a maximum-matching algorithm on a graph having n vertices. Future work will focus on how to incorporate user feedback into the model so that members themselves assist in finding a good matching in a distributed manner, as to some extent is typically done now in social networks, as well as incorporating a rewards mechanism into the system, and examining fuzzy multi-dimensional matchings.

ACKNOWLEDGMENTS

Thanks to Anthony Delli Colli and Jim Burns of Elbrys Networks, Inc. for numerous helpful discussions about this work.

REFERENCES

[1] Power M.L. and Schulkin J., *The Evolution of Obesity*. Johns Hopkins University Press, 2009.

[2] Ardigo D., et al. Hyperinsulinemia predicts hepatic fat content in healthy individuals with normal transaminase concentrations, Metabolism, 2005; **54**(12): 1566-70.

[3] Farin H.M., Abbasi F. and Reaven G.M., Body mass index and waist circumference both contribute to differences in insulin-mediated glucose disposal in nondiabetic adults. *American Journal of Clinical Nutrition*, 2002; **83**(1): 47-51.

[4] Reaven G., All obese individuals are not created equal: insulin resistance is the major determinant of cardiovascular disease in overweight/obese individuals. *Diabetes and Vascular Disease Research*, 2005; **2(3)**:105-112.

[5] Reaven G., Abbasi F. and McLaughlin T., Obesity, insulin resistance, and cardio-vascular disease. *Recent Progress in Hormone Research*, 2004; **59**: 207-223.

[6] National Health Expenditure. CMS; Bureau of Economic Activity, 2009.

[7] Piniewski B., Technology-enabled health impact measures, presentation, 2009.

[8] Connecting Wellness Communities. Elbrys Networks, Inc. www.elbrys.com, 2010.

[9] Agnarsson G. and Greenlaw R., *Graph Theory: Modeling, Applications, and Algorithms*. Prentice Hall, 2007.

[10] Garey M.R. and Johnson D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.

[11] Yannakakis M. and Gavril F., Edge dominating sets in graphs, unpublished manuscript, 1978.

[12] Itai A. and Rodeh M., Some matching problems. *Automata, Languages, and Programming*. Lecture Notes in Computer Science, Springer, Berlin, 1977; **52**: 258-268.

[13] Karp R.M., Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher (editors), *Complexity of Computer Computations*, Plenum Press, New York, 1972: 85-103.

[14] Dobkin D. and Ladner R., Private communication cited in [11], 1978.

[15] Micali S. and Vazirani V.V., An $O(\sqrt{V} \cdot E)$ algorithm for finding a maximum matching in a general graph. In *Proceedings of the 21st Annual IEEE Symposium on the Foundations of Computer Science*, 1980: 17-27.

[16] Vazirini V.V., A Theory of alternating paths and blossoms for proving correctness of the general graph matching algorithm, *Combi-natorica*, 1994; **14**: 71-91.

[17] Mucha M. and Sankowski P., Maximum Matching via Gaussian Elimination. In *Proceedings of the 45th IEEE Symposium on the Foundations of Computer Science*, 2004: 248-255.

[18] Edmonds J., Maximum matchings and a polyhedron with 0,1-vertices, *Journal of Research of the National Bureau of Standards*, 1965; **69B**: 125-130.

[19] Gabow H.N., An efficient implementation of Edmond's algorithm for maximum matchings on graphs, *Journal of the Association for Computing Machinery*, 1976; **23**(2): 221-234.

[20] Lawler E.L., *Combinatorial Optimization: Networks and Matroids*. Holt, Rienhold, and Winston, 1976.

[21] Gabow H.N., Data structures for weighted matching and nearest common ancestors with linking, In *Proceedings of the 1st Annual SIAM Symposium on Discrete Algorithms*, 1990: 434-443.

[22] Lov¿sz L. and Plummer M.D., *Matching Theory*, volume 121 *of Annals of Discrete Mathematics*, North Holland, 1986.

[23] Karp R.M., Upfal E. and Wigderson A., Constructing a perfect matching is in Random NC. *Combinatorics*, 1986; **6**(1): 35-48.

[24] Mulmuley K., Vazirani U.V. and Vazirani V.V., Matching is as easy as matrix inversion. *Combinatorica*, 1987; **7(10)**: 105-113.

[25] Dahlhaus E., Hajnal P. and Karpinski M., Optimal parallel algorithms for the Hamiltonian cycle problem on dense graphs. In *29th Annual IEEE Symposium on Foundations of Computer Science*, 1988: 186-193.

[26] Uehara R. and Chen Z., Parallel approximation algorithms for maximum weighted matchings in general graphs, *Information Processing Let- ters*, 2000; **76(1)**: 13-17.

[27] Cook S., Personal communication, 1982.

[28] Mayr E.W. and Subramanian A., The complexity of circuit value and network stability, *Journal of Computer and System Sciences*, 1992; **44(2)**: 302-323.

[29] Greenlaw R., Hoover H.J. and Ruzzo W.L. *Limits to Parallel Computation: P-Completeness Theory*, Oxford University Press, 1995.

[30] Gibbons, A. and Rytter, W. *Efficient Parallel Algorithms*, Cambridge Uni- versity Press, 1988.

APPENDIX

A Computational-Complexity Theory Background

In this appendix we present the basics of *computational-complexity theory* needed for an understanding of our results, occasionally oversimplifying a little to avoid technical details. Computational complexity involves the study of the computing resources required to solve problems on computers. For example, we can ask how many computational steps

are needed to determine the minimum of n numbers. (The letter n is often used in computer science to denote the size of the input to a problem.) There is an obvious program that accomplishes this task using one "loop" in $O(n)$ steps. Here the big-oh notation means roughly to within a constant times the value contained inside the parentheses. Thus, this problem can be solved on a sequential computer (for example, a desk-top computer with sufficient memory) in a time that scales as a (linear) *polynomial* in the size of the problem. The class of all such problems is denoted P (for polynomial). It is generally acknowledged that any problem in P has a tractable solution in the sequential computing domain. It is easy to see that the problem of sorting n numbers is in P by invoking the loop to find the minimum on n numbers n times, resulting in an $O(n^2)$ algorithm for sorting.

On the other hand, there are problems for which no polynomial-time algorithm is known. The well-known class of NP-complete problems are almost certainly not in P. They are considered intractable. Examples of such problems come from many different disciplines [10]. The principle question in the parallel computing domain is whether one can obtain a "significant speed-up" for every problem in P by using a *parallel computer.* Although there are formal mathematical models that define the concept of parallel computer precisely, here simply think of many computers all working together to try to solve a given problem. The phrase significant speed-up will be made technically precise shortly.

Consider the trivial problem of randomly assigning a 0 or 1 to every site of a lattice. Such a task requires $O(n)$ steps on a sequential computer but if each lattice site is assigned a processor (equipped with a random number generator) then the task can be carried out in constant parallel time, denoted by $O(1)$. As a second example, consider again the problem of computing the minimum of a list of distinct numbers $x1,..., xn$. In parallel, we may pairwise compare "neighbors" using $n/2$ processors. The $n/2$ minima are then compared pairwise and so on until the global minimum is found. It is clear that this procedure requires $O(\log n)$ parallel steps and $n/2$ processors. In each of our simple examples we went from an $O(n)$ time algorithm to a much faster parallel algorithm. It is not at all clear that similar dramatic speed-ups are possible for the dynamic-matching problems discussed in this work.

In what follows we use the conventional PRAM model [30] of parallel computation: a collection of processors distinguished by numerical labels each run the same program and communicate through a common global random-access memory. The acronym pram stands for *parallel random-access machine.* It is assumed that any processor may address any memory element in a single time step. This notion of parallel time is idealized and ultimately unphysical due to finite-signal propagation speeds. Nonetheless, parallel time reflects a fundamental aspect of computation since such models can be simulated efficiently on more realistic parallel machines.

Parallel time on a PRAM is essentially equivalent to "logical depth": the minimum number of logical operations that must be carried out in sequence to complete a computation. This concept can be made rigorous by considering families of Boolean circuits (see [29]). A Boolean circuit is a feed-forward network of logical gates with n inputs and a single output. The *size* of a circuit is the number of gates and the *depth* of the circuit is the length of longest path from an input to the output. A family of Boolean circuits, one for each problem size, can simulate a PRAM programmed to solve a given problem and vice versa (see [29] for a description of such a simulation). Complexity

classes defined by PRAMs can be equivalently defined in terms of *circuit families*. *Parallel time* on a PRAM scales in the same way as circuit depth, and the total number of operations carried out by the PRAM scales in the same way as circuit size.

We now define another computational-complexity classes of interest in this paper (see [29] for a more in-depth discussion). The class NC consists of prob- lems that can be solved in poly-logarithmic ($O(\log^k n)$) parallel time using a number of processors bounded by a polynomial in the input size n. Here k denotes a constant. It is known that NC $\subseteq$ P since a polynomial number of processors that operate for only a poly-logarithmic amount of time (or for that matter a polynomial amount of time) can be directly simulated sequentially in polynomial time. As previously mentioned, the problem of finding the minimum of n numbers is in NC, as is the problem of sorting n numbers. Hundreds of other important computational problems have been shown to be in NC sparking a wide interest in this class.

Strictly speaking, both NC and P refer to *decision problems*, that is problems whose output on a fixed input is either "no" or "yes." But, sometimes researchers speak informally about *search* problems being in these classes as well. One of the most basic questions in parallel computation is whether all problems with efficient sequential solutions have fast parallel solutions?, that is, is P = NC? Although the P versus NC question has not been settled, it is widely believed there are problems in P that require more than poly-logarithmic parallel time. A problem L is *P-complete* if the following holds: L is in P and if a poly-logarithmic time, polynomial-processor algorithm for L existed it could be used as a subroutine to solve *any* problem in P in poly-logarithmic time using only a polynomial number of processors. The P-complete problems are the computationally most difficult problems in the class P. Under the assumption that NC does not equal P, problems that are P-complete are in P but not in NC. Thus, P-complete problems are not solvable in parallel in poly-logarithmic time using a polynomial number of processors, unless NC = P (which is considered as unlikely as P = NP).

The computational-complexity class NP mentioned in the abstract and the introduction is the class of problems whose solutions can be guessed and verified for correctness in polynomial time. For example, if we guess a series of cities in an instance of the TRAVELING SALESMAN PROBLEM, it is easy to check in polynomial time if the sum of the distances between them is less than the given bound b. A problem L is *NP-complete* if the following holds: L is in NP and if a polynomial-time algorithm for solving L existed then it could be used as a subroutine to solve any problem in NP in polynomial time. The NP-complete problems are the computationally most difficult problems in the class NP. These problems are considered intractable meaning that they do not have any practical algorithms for solving them; and furthermore, they never will, unless P = NP. The P versus NP question is considered one of the ten most important open problems in all of computer science and mathematics.

A large class of problems are P-complete; a catalog of 500 such problems is given in [29]. It is interesting to note that one method for proving P-completeness is closely related to the notion of *computational universality*. A specific P-complete problem, namely the CIRCUIT VALUE PROBLEM, is to evaluate the output of an arbitrary Boolean circuit with given inputs. Thus one way to prove that a problem L is P-complete is to show that an algorithm for L could be used to simulate an arbitrary Boolean circuit on a fixed input.