SCHEDULING INTERVIEWS FOR A JOB FAIR

John J. Bartholdi, III * Kevin L. McCroan †

April 4, 1989; revised April 3, 2003

Abstract

We describe the design and use of a program to schedule job interviews for law firms and students at a job fair. It has been used to manage the Southeastern Public Interest Job Fair for the last five years. The program uses a new scheduling algorithm that produces particularly convenient schedules.

1 Introduction

During the last five years the Southeastern Public Interest Job Fair has been managed by means of a computer program that we wrote. In addition to handling many other administrative chores, the program schedules all job interviews so that there are no conflicts. The algorithm on which the program is based is apparently new. It is simple to state and interesting to analyze. Moreover, it produces particularly convenient schedules because of circumstances special to this job fair.

*This was *pro bono* work done while the first author was supported in part by a Presidential Young Investigator Award from the National Science Foundation (ECS-8351313), and by the Office of Naval Research (N00014-85-K-0147). Address: School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332

†Supported in part by the Office of Naval Research (N00014-85-K-0147). Address: School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332

2 An unusual job fair

The Southeastern Public Interest Job Fair is held each year during a weekend in November, when 25–50 law firms from all over the U.S. and 100–200 law students from all over the Southeast come to Atlanta for job interviews. To help make this market orderly and efficient, we provided software to schedule meetings between law firms and students. The challenge was to schedule without conflicts all of the meetings requested by the participants.

This job fair is unusual in that it is oriented towards employers in public interest law, such as public defense or legal services for the poor. The job fair was created to help overcome the difficulties in recruiting faced by public interest law firms. Such employers generally have little money to spend on independent recruiting, and other job fairs do not address their concerns. For example, public interest law firms generally do not find likely job candidates at the corporate job fairs, which attract a different type of student and which are also rather expensive.

For more complex reasons job fairs sponsored by the placement centers of law schools tend to be unsatisfactory. First, they are not effectively managed. A placement center is likely to be overwhelmed with the logistics of a job fair, especially the scheduling of hundreds of interviews. In fact, it is typical that an employer not know whom he is to meet, or when, until the job fair begins. Even then the employer might learn his schedule only as it happens.

Another problem is that a job fair run by a placement center is generally managed primarily for the convenience of the placement center itself and then for the students. The needs of the employers seem not to be a priority. The goal of a placement center is to market its students, and it does this by getting them interviews. A placement center typically apportions interview slots according to its own priority system, which can seem quite arbitrary to an employer. For example, employers are generally not allowed to screen applicants beforehand. An employer must interview anyone the placement center schedules, quite independently of their qualifications for the job.

In addition, there are often arbitrary restrictions imposed by a placement center. For example, some job fairs are jointly sponsored by several schools in the same city. It is typical for them to apportion the interviews equally among the students of each school, so that a school perceived as weaker will not be embarrassed by getting fewer interviews for its students. Unfortunately, this means that an employer can be restricted to interview at most a few students from each school. For example, we attended a job fair that required each employer to interview 2 students from each of four schools. One employer was sought out by 8 interested students from the same school, and none from the other schools. Nevertheless, that employer was allowed to interview only 2 of the students, and had to sit idle for the rest of the day, despite the fact that 6 other students wanted interviews! Because of such restrictions a law firm might be lucky to interview even a few students who meet the straightforward qualifications for the job. This becomes more than an annoyance since most law firms travel from out of town, with consequent expenses for transportation, meals, and hotel.

Another difficulty with the job fairs of placement centers is that the interview times are quite short, generally only 15–20 minutes. Placement centers schedule short meetings so that more students can get interviews, but such short meetings are not useful for the law firm or the student. This is especially difficult on public interest firms, who must judge whether the candidate has the idealism and devotion to stick to a low-paying public service job. While corporate employers invite prospects for further interviews, public interest employers have such limited funds that they can rarely afford this.

Partly in reponse to these difficulties, the Southeastern Public Interest Job Fair allows both the students and the law firms to screen each other beforehand based on job descriptions and résumés, and then makes it as simple as possible for participants to talk when there is mutual interest. In addition, for the convenience of all, this job fair provides each participant with a copy of his interview schedule several weeks in advance.

3 Planning for the job fair

The Southeastern Public Interest Job Fair synchronizes all meetings to take place at common times, for example, 08:30–09:00, 09:05–09:35, and so on. There are 13 meeting times on Saturday and 10 on Sunday, so each participant can have up to 23 meetings. Accordingly, each person is allowed to request up to this many meetings through the following process. Several months in advance, the job fair solicits law firms to attend. Then it solicits students by sending résumés of participating law firms and their job descriptions to all the law schools in the Southeast. Interested students select up to 23 employers with whom they would like to interview. Each student then sends copies of his résumé back to the job fair, which sorts them and forwards them to the appropriate law firms. Each law firm screens its applicants and selects up to 23 to interview. The law firm then notifies the job fair so that the meetings can be scheduled.

Despite the large fixed (travel) cost of attending the job fair, a typical student requests only a few interviews. (At the 1988 job fair almost a quarter of the students requested only a single interview.) This is partly because a student is unlikely to be committed already to a career in public interest law. He is more likely to be exploring options rather than single-mindedly searching for a job. On the other hand, the law firms have a definite goal in attending the job fair: they want to hire lawyers. A typical law firm selects 8–16 students to interview (average for the 1988 job fair: 11.7).

Generally a student does not get all the interviews he requests (average for the 1988 job fair: 3.7 granted out of 4.6 requested). This is usually because of low class rank or the lack of demonstrated commitment to a career in public interest law (for example, no volunteer work). Students are also rejected for more banal reasons, such as failure to submit a complete résumé.

It is theoretically possible that all the law firms want to meet the same small set of students, which would make for a rather narrow job fair. One way of avoiding this would have been to require each participant to submit a rank order of all prospective partners that they find acceptable, and then to construct a "sta-

ble matching" to determine who is to meet whom [5]. However, the managers of the job fair felt that this would be an unnecessary and unwanted complication. They felt that it would be especially difficult to extract preferences from the law firms since they do not want to make such micro-decisions based only on résumés. In public interest law firms the personnel committees are usually composed of lawyers who have frenetic case schedules, and for whom hiring is an additional responsibility. They simply would not submit such detailed lists. (Even now, it can be difficult just to find out whom the law firms want to meet. The scheduling is invariably delayed several days while the managers telephone the last several law firms to get their lists.)

In any event, the diversity of the participants ensures that the job fair will never be pathologically narrow. The law firms have included those with 2 lawyers and those with 150 lawyers; those that practice criminal law and those that practice civil law; those in rural Mississippi and those in downtown Atlanta; a district attorney's office in Brooklyn and the Environmental Protection Agency. This attracts a similarly diverse group of students.

After the students and then the law firms have chosen, the job fair knows who is to meet whom. The difficulty is to produce a schedule that is free of conflicts. Other job fairs, such as those sponsored by placement centers, schedule naïvely, by simply assigning times to meetings until there is a conflict. If there is not an obvious way to resolve the conflict by rescheduling one of the participants, the requested meeting is discarded as "not possible". However, scheduling like this can, in the extreme case, be disasterous for a participant. For example, imagine scheduling the last law firm. It is possible that all 23 students they want to meet are available only at 09:00, in which case only 1 of 23 intended meetings would be scheduled.

We tried to estimate the number of discarded meetings at two placement center job fairs. However, the scheduling process was too *ad hoc* to permit estimates. Moreover, the scheduling had apparently been done by several people independently of each other. All that we could determine was that "some" meetings had been deemed not possible.

It is a goal of the Southeastern Public Interest Job Fair to schedule every requested meeting. This is important to the participants because career decisions are important, and because most people incur a large fixed cost to travel to the job fair. Since participants have screened each other, interviews in the Southeastern Public Interest Job Fair are generally more serious than in other job fairs, where people are likely to be "just shopping".

Our computer program guarantees that all requested meetings will be scheduled without conflicts. Because the program does this accurately and conveniently, the Southeastern Public Interest Job Fair, unlike others, is able to send every participant a copy of his schedule several weeks before the fair.

4 Structure of the program

The program maintains two separate lists, one of law firms and one of students, and the user can move freely between lists. The program provides the ability to browse through each list as if the information about each participant were written on an index card, and the cards arranged in alphabetical order. The user can move among the cards by pressing the cursor keys. On the "front" of each card is registration information about the participant, and on the "back" is their schedule. The user can "turn" the cards over with a keystroke.

Registration information includes fields like name and address. There is also a note pad for the job fair manager to write reminders, such as phone calls to return or special requests. In addition, for each law firm there is a field in which can be entered a message which will be printed on the schedules of all students to meet this law firm. For example, a law firm can remind all of its interviewees to bring a writing sample to the interview.

When viewing the registration information, keystrokes affect only registration. For example, pressing the Insert key inserts a blank index card for the user to register another person to attend the job fair; similarly, pressing the Delete key deletes the currently-viewed person and all of his meetings from the job fair.

After the user turns the card over to display the schedule of the current person, then keystrokes affect only the schedule. For example, pressing the Insert key allows the user to schedule a new meeting for the current person: a small window pops open in which the user can scroll among the participants from the other list; when the correct one is found, pressing the Enter key instantly schedules the meeting. The user simply indicates that these two participants are to meet and the program schedules the meeting.

5 Scheduling meetings

5.1 A new scheduling algorithm

To schedule meetings between law firms and students one must assign a time to each meeting so that no participant has more than one meeting scheduled at any one time. This is known as a "time-tabling" problem, which in this case can be modelled as a problem in graph theory known as "edge-coloring a bipartite graph". (See [1] for a survey of time-tabling, and [2, 8] for related results.) The model is as follows. Consider the bipartite graph G that has one set of vertices corresponding to law firms and the other set corresponding to students; in addition, let the set E of edges correspond to intended meetings. Now assigning times to meetings so that there are no conflicts is the same as assigning colors to edges so that no two edges that meet at a common vertex have the same color. Since no two edges incident to the same vertex can bear the same color, to color the edges of a graph requires at least as many colors as the maximum degree of any vertex. In fact, this many colors is also sufficient for a bipartite graph [6, for example]. Our algorithm proves this result (and in fact strengthens it slightly), but has other, more interesting virtues. In particular, it is simple, and in the context of scheduling, it is quite natural. Furthermore, our algorithm will be seen to generate especially convenient schedules for the job fair.

Our algorithm assigns an abstract time from $1, 2, \ldots$ to each meeting. To

```
PROCEDURE Reschedule(
  person : PersonType;
  meeting: MeetingType );
BEGIN
  Let otherPerson be he who is to meet person in meeting;
  Let time be the first time at which person is available;
  Schedule meeting for time;
  IF this causes a conflict for otherPerson THEN
    Let otherMeeting be the conflicting meeting;
    Reschedule( otherPerson, otherMeeting );
  END;
END Reschedule;
```

Figure 1: This algorithm reschedules *meeting* for the earliest time at which *person* is available; then it calls itself to resolve any conflicts this might have introduced.

schedule a meeting between persons u and v, it sets the time of the meeting to the earliest time at which one of the people, say u, is available. If this causes a conflict for v (because he has another meeting at that time), the algorithm reschedules the other meeting for the first time that v is available; and continues resolving successive conflicts by rescheduling meetings at the first available time. This is stated more formally in Figure 1.

We call this algorithm RESCHEDULE since, whenever a conflict is introduced by scheduling a meeting, the algorithm reschedules the conflicting meeting to resolve the conflict. The sequence of meetings rescheduled during an invocation of the algorithm traces a directed path along the corresponding edges of the graph. The direction of the path is given by the sequence in which the vertices are encountered while rescheduling.

Rescheduling along a path is part of a more general idea in graph theory known as "augmenting paths": to extend a structure (such as a matching, edge-coloring, or schedule), add a new piece and resolve conflicts locally, along a path. An edge-coloring algorithm based on this idea is implicit in the standard proof that a bipartite graph is edge-colorable with $d_{\max}$ colors, where $d_{\max}$ is the largest degree of any vertex [6]. This algorithm is made explicit in [4], where it is called algorithm AUGMENT. When interpreted for scheduling, algorithm AUGMENT works as follows. To schedule a meeting between law firm u and student v determine a time t_u at which u is available and schedule the meeting for t_u; if this causes a conflict for v, determine a time t_v at which v is available, and reschedule the conflicting meeting for t_v. Continue rescheduling conflicts alternately for t_u or t_v. It is straightforward to argue that an invocation of this procedure reschedules no more than n meetings.

Algorithm AUGMENT differs from ours in that it chooses two arbitrary times and reschedules any conflicting meetings for one of those two times. This is reasonable in the context of edge-coloring, but misses the additional structure of time-tabling problems, where the times have a natural (clock) order. Our algorithm exploits this structure by always rescheduling for the *earliest* available time. As we will show, this produces convenient schedules.

An interesting feature of our algorithm is that meetings can appear more than once in a rescheduling path, so that they are *re*-rescheduled. Thus at first glance it is not clear that algorithm RESCHEDULE terminates. It does, however, and quickly. In fact, we will show that a rescheduling path contains no more than $4|E|$ meetings, counting duplicates. Any single meeting can appear on the rescheduling path many times, or many different meetings can appear a few times each—but the total will not be "too large".

We begin by establishing some technical results.

Lemma 1. *Let t_i be the times assigned to successive meetings along a rescheduling path; then $t_i \neq t_{i+1}$ and $t_i \geq t_{i+2}$.*

Proof. The first property holds because successive meetings are rescheduled to

resolve conflicts.

To establish the second property, let $\{(v_i, v_{i+1})\}$ be the sequence of meetings rescheduled by the algorithm. When the meeting (v_i, v_{i+1}) is scheduled for time t_i, this must have conflicted with meeting (v_{i+1}, v_{i+2}), so that meeting (v_{i+1}, v_{i+2}) must have been previously scheduled for time t_i. When the algorithm reschedules this meeting for time $t_{i+1} \neq t_i$, then person v_{i+2} becomes available during time t_i; and since the algorithm schedules for the first available time, the meeting (v_{i+2}, v_{i+3}) will be scheduled for time $t_{i+2} \leq t_i$. □

By Lemma 1 the sequence of times assigned along the rescheduling path can be partitioned into non-increasing subsequences $T_{\text{odd}} = \{t_{2k-1}\}$ and $T_{\text{even}} = \{t_{2k}\}$. Furthermore, since the graph is bipartite, the times of one subsequence are those assigned when scheduling for the convenience of law firms; and the times of the other subsequence are those assigned when scheduling for the convenience of students.

Definition When $t_i > t_{i+2}$ for some i in a rescheduling path, then we say that the corresponding subsequence has been *reduced*.

Lemma 2. *At least one of the subsequences T_{odd} or T_{even} must have been reduced between successive appearances of any person along the rescheduling path.*

Proof. Consider any cycle $\ldots, v_i, v_{i+1}, \ldots, v_{i+k} = v_i, \ldots$ on a rescheduling path, and let the algorithm schedule meeting (v_i, v_{i+1}) for time t_i and (v_{i+1}, v_{i+2}) for t_{i+1}. Suppose that neither subsequence is reduced between successive appearances of v_i on the path. Then for meeting (v_{i+k-1}, v_i) the algorithm must either replace time t_i with t_{i+1} or else replace t_{i+1} with t_i. If it replaces t_i with t_{i+1}, then it could not have scheduled (v_i, v_{i+1}) for time t_i since v_i would not have been available at that time. If instead, the algorithm replaces t_{i+1} with t_i, then the cycle must contain an odd number of edges, which contradicts the fact that the graph is bipartite. In either case there is a contradiction, and so the claim must hold. □

Finally, we have the key result. Let the degree of vertex v be d_v, which represents the number of meetings to be scheduled for person v.

Lemma 3. *Person v cannot appear more than $2d_v$ times on any given rescheduling path.*

Proof. For a given rescheduling path, consider any pair of successive appearances of person v. Without loss of generality, assume that the first appearance of v was occasioned by rescheduling a pair of meetings for times t_i and t_{i+1}, and the second appearance was occasioned by rescheduling a pair of meetings for times t_{i+k} and t_{i+k+1}. From Lemma 2, it must be that $t_{i+k} < t_i$ or $t_{i+k+1} < t_{i+1}$. Therefore one of the following cases must hold.

Case 1 $\min\{t_{i+k}, t_{i+k+1}\} < \min\{t_i, t_{i+1}\}$

Case 2 $\max\{t_{i+k}, t_{i+k+1}\} < \max\{t_i, t_{i+1}\}$

In the first case, since the algorithm schedules at the earliest convenience, $t_{i+1} \leq d_v$, so that $\min\{t_i, t_{i+1}\} \leq d_v$ and $\min\{t_{i+k}, t_{i+k+1}\} < d_v$. Since both sequences are non-increasing, Case 1 can happen no more than d_v times.

In the second case, the meeting of v scheduled for time $\max\{t_i, t_{i+1}\}$ cannot appear again on this rescheduling path. This follows from noting that $\max\{t_i, t_{i+1}\} > \max\{t_{i+k}, t_{i+k+1}\}$; and because the subsequences are non-increasing, no subsequently rescheduled meeting can conflict with this meeting. Therefore the cycle has determined a unique meeting of v that cannot appear again in this rescheduling path. Consequently Case 2 cannot happen more than d_v times. □

Our main theorem follows immediately.

Theorem 1. *No more than $4|E|$ meetings are rescheduled in any single invocation of the algorithm.*

Proof. Since person v appears no more than $2d_v$ times in a rescheduling path, the total length of the path cannot exceed $\sum 2d_v = 4|E|$. □

Since there are $|E|$ meetings and each invocation of the algorithm schedules one more of them, the worst-case effort to schedule all meetings is $O(|E|^2)$. For applications of this type the number of participants might become large, but the number of meeting times is likely to remain bounded by a small constant. Therefore it is probably more accurate to interpret this complexity as $O(n^2)$, where n is the total number of participants in the job fair.

When meetings were batch-scheduled, the computational effort was perceptible (a few seconds). For this and other reasons we hid the effort from the user by distributing it throughout the data entry. The user never sees anything explicitly about scheduling in the program. Instead, he simply indicates that two people want to meet, and then the program invokes algorithm RESCHEDULE to schedule the requested meeting. Thus scheduling is done incrementally and is indistinguishable from data entry. At all times the program maintains a conflict-free schedule of all meetings requested so far. This simplification allows naïve users to remain oblivious to scheduling.

Our program implements RESCHEDULE as follows. The user looks at the schedule of, for example, a law firm and presses the Insert key, whereupon an auxiliary window pops open to allow the user to scroll through the alphabetical list of students. When the user finds the student to meet that law firm, he presses the Enter key to indicate that a meeting is intended. The program then schedules that meeting at the earliest time at which the law firm is available. If the student already has a meeting scheduled at that time, the program reschedules it to the earliest time at which the student is available, and so on.

There are several alternative algorithms for edge-coloring that have better asymptotic worst-case performance than RESCHEDULE. For example, Gabow suggested an algorithm that uses a divide-and-conquer strategy to achieve a worst-case running time of $O(n^{1/2}|E| \log n + n)$ [3]. This approach was subsequently extended in [4] to give algorithms of complexity $O(|E|(\log n)^2)$ and $O(n^2 \log n)$. However, for this application there is no reason to use one of the faster-in-the-worst-case algorithms and several reasons not to. The most telling objection is that these algorithms are considerably more complicated than ours.

The possible gain in speed is not worth the effort of coding and debugging. (In contrast, it is hard to imagine a simpler implementation than the 7 lines of code for RESCHEDULE.) In addition, since there is no natural way to invoke the more complicated algorithms incrementally as the data is entered, they must be run as batch procedures and therefore might actually seem to the user to require *more* time than our current implementation, which appears to be instantaneous, even on the slowest computer. Finally, as we show in the next section, RESCHEDULE can guarantee convenient schedules, and we do not know that for the alternative algorithms.

5.2 Individual convenience versus the social good

The job fair schedules meeting times for the social good, not for the convenience of any particular individual. It is because the program does not guarantee a convenient schedule for any individual that it can guarantee that all meetings will be scheduled. Unfortunately, individual convenience and the scheduling of all meetings are inconsistent goals, and it is impossible to guarantee both. This has annoyed a few students who submitted elaborate requests as to how their meetings should be scheduled. Because of the *ad hoc* scheduling in other job fairs, they apparently perceived convenience and fairness to be independent, if they thought about it at all.

The most frequent request has been that an individual's meetings be scheduled consecutively. This cannot be guaranteed for all participants, as can be seen when each of two law firms are to meet each of three students. In this case, it is impossible for each student to have his meetings scheduled consecutively.

While we cannot guarantee convenience for individuals, the program has been able to provide a weaker sort of convenience, a convenience for the group: that the job fair schedule "more meetings sooner" so that people tend to finish all their meetings early.

The convenience of "more meetings sooner" can be idealized as maximizing the cumulative number of meetings by each time period; that is, by having

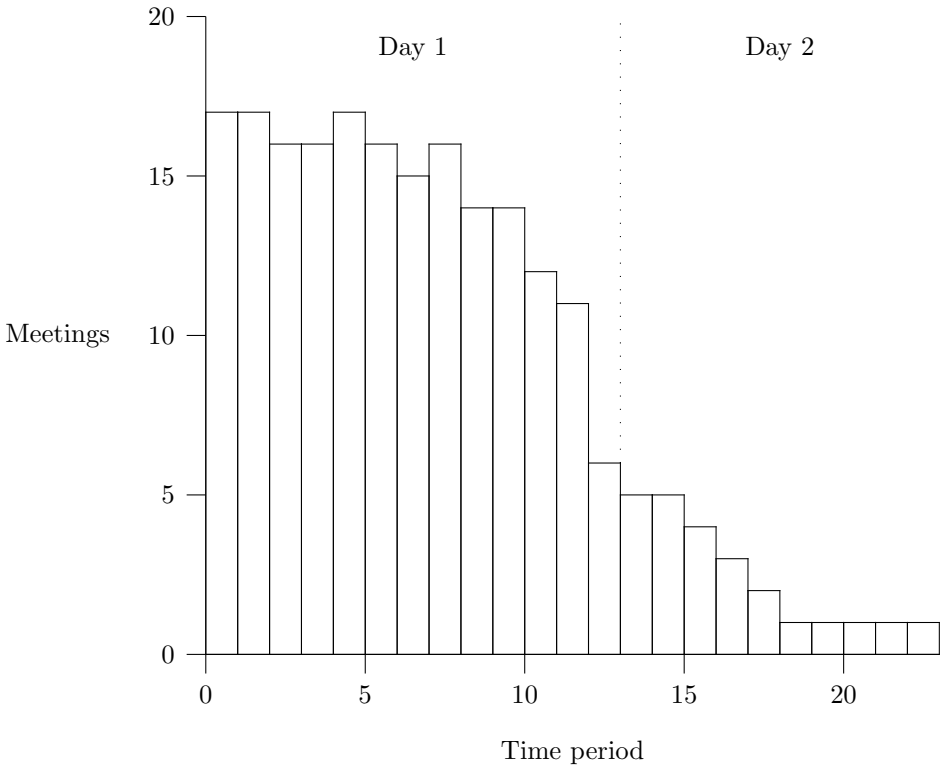

Figure 2: Distribution of meetings for the 1988 job fair. Most meetings were held on the first day.

as many meetings as possible during the first time period, as many meetings as possible during the first two time periods, and so on. Unfortunately, as shown in [8], no scheduling algorithm can guarantee this. However, in practice RESCHEDULE has performed extemely well, as illustrated by the 1988 job fair. This was an unusually small job fair, but it is otherwise representative. Figure 2 shows that most meetings were held on the first day of the job fair.

How can this good performance be explained? It is attributable to a circumstance that is special to this application: that a typical law firm has been rather more heavily-booked than a typical student. Thus if a law firm has free time early in the job fair, some one of the many students it is to meet—who are relatively lightly-booked—is also available, and our algorithm would schedule

such a meeting early in the job fair. As a result, nearly all law firms have had schedules with consecutive meetings, and so in each successive time period, (almost) as many meetings as possible have been scheduled. The following gives a more formal explanation.

Lemma 4. *Algorithm* RESCHEDULE *schedules each meeting* (u, v) *for a time no greater than* $\max(d_u, d_v)$.

Proof. Whenever the algorithm schedules or reschedules meeting (u, v), it is either at the earliest time at which u is available, or at the earliest time at which v is available. But u must be available within the first d_u time periods, and v must be available within the first d_v time periods. □

This strengthens the standard result on edge-coloring that all meetings can be scheduled for times no greater than $d_{\max}$. (In fact, our algorithm can give a still stronger version of this result, as we show in the Appendix.) However, the following corollary is of more immediate interest to us. It says that if law firms are more heavily-booked than the students they meet, then the algorithm will schedule "more meetings sooner".

Theorem 2. *If no student has more meetings than the law firms he is to meet, then algorithm* RESCHEDULE *maximizes the cumulative number of meetings by each time period.*

Proof. For each student s to meet law firm l, by hypothesis $d_l \geq d_s$, and so their meeting will be assigned a time within the first d_l time periods. But since there are exactly d_l such meetings for l, and since none of them can be assigned the same time, law firm l must be booked for consecutive meetings beginning at the first time period. Finally, since every law firm is booked for consecutive meetings beginning at the first time period, by each time period, as many meetings as possible have been held. □

The hypothesis of this theorem idealizes somewhat the circumstances of the job fair. Generally it has been true only that *most* law firms had more meetings

than *most* of the students they were to meet. Nevertheless, the conclusion of the corollary continues to hold, at least approximately, as shown by the following thought experiment. For law firm l with fewer meetings than the busiest student s it is to meet, let its deficit be $d_s - d_l$, and for each such law firm register $d_s - d_l$ imaginary students for the job fair and schedule l to meet them. Now the conditions of the theorem hold for this job fair augmented by imaginary students, and so each law firm will be scheduled for consecutive meetings beginning at the first time period. However, meetings with imaginary students correspond to idle time, which can be interruptions in the consecutivity of meetings. Therefore, the fewer imaginary students necessary—that is, the more nearly the hypothesis of the corollary holds—the more nearly consecutive will be the meetings of each law firm.

In fact, the 1988 job fair very nearly maximized the cumulative meetings by each time period since, as shown in Figure 3, most law firms requested more meetings than most students, and so most law firms had all their meetings scheduled consecutively. Because of this, most people completed all their meetings on a single day. (In contrast, if meetings had been scheduled "randomly", then most people would have had meetings on both days.)

There is a possible cost to providing the convenience of "more meetings sooner": the job fair must provide sufficient meeting rooms for the busiest time period (in this case, a room for every law firm). However, in the job fairs we have attended there have been no realistic limits on meeting space, since if there are too few rooms, the managers simply arrange a table and chairs in a corner. An early version of our program included the ability to reschedule meetings to level the demand for rooms, but since this capability was never needed, we removed it.

5.3 Adjustments to the schedule

It is a property of our algorithm that the schedule it produces depends on the sequence in which the meetings are considered. This has been useful in the

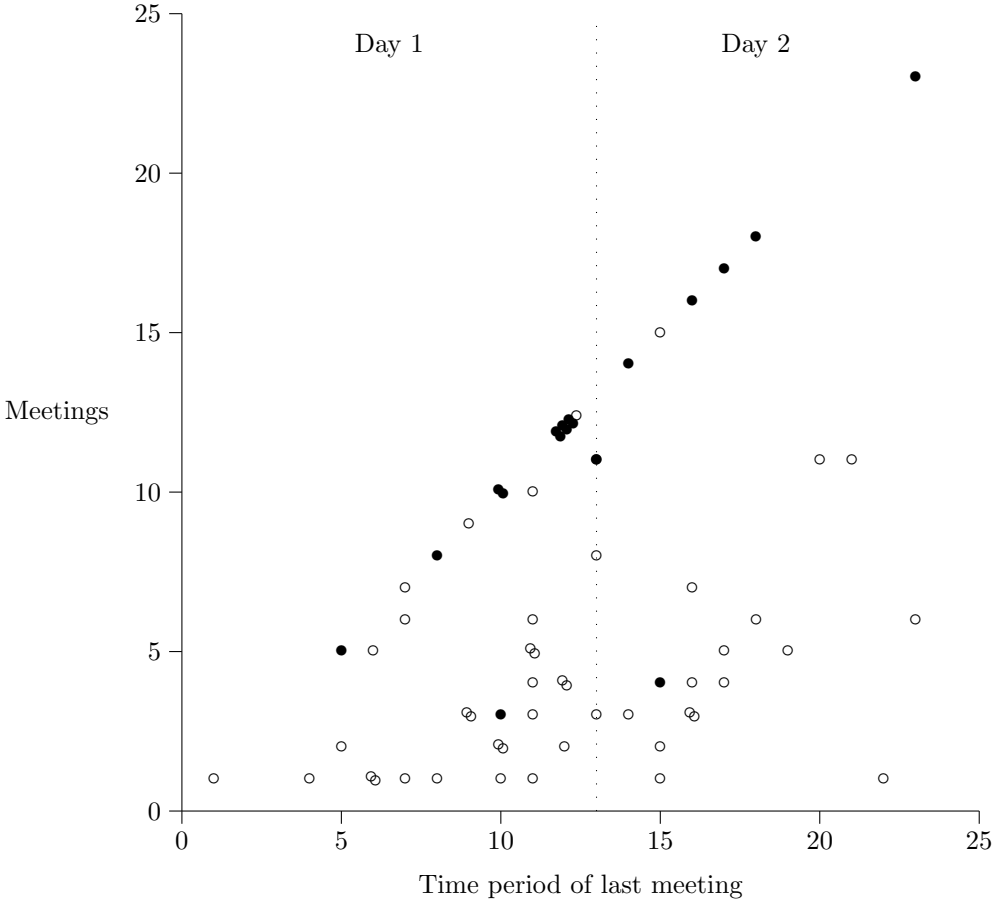

Figure 3: Each "•" corresponds to a law firm and each "○" corresponds to a student. Most law firms appear on the diagonal, which means that they were scheduled for consecutive meetings.

following way. The job fair has always received several requests for special scheduling. Since such requests have been few, we have been able to satisfy them without inconveniencing others. We have done this by building special schedules last, so that they are unlikely to be revised by subsequent scheduling. For example, some people have asked to not have meetings on the first morning of the job fair because of late-arriving flights. For each such person we register a "dummy" participant, and then schedule them to meet this dummy at each of the forbidden times. Then when we schedule their actual meetings, none is scheduled at the times reserved for the dummy meetings. Finally we delete the dummy participant, leaving the schedule free in the morning. (Of course, this technique cannot be guaranteed to work if too many people make such requests, since then it might happen that previously-accommodated people will have meetings rescheduled into their forbidden times.)

Some people have asked to have no meetings scheduled on the last afternoon of the job fair. No rescheduling has been necessary yet to meet such requests since, as discussed previously, only the busiest people have meetings at the end of the job fair.

(We note that another argument against using more complicated scheduling algorithms is that the user loses the immediate control that is essential to adjust a schedule. We were able to handle these exceptional cases effectively because our algorithm constructs a schedule incrementally. Furthermore, since our algorithm is simple, the user can anticipate its effects on the job fair (at least locally) and therefore control it to his advantage.)

Other than wishing to provide convenience when possible, the managers of the job fair do not care about secondary objectives. They want to print schedules and be done with it.

6 Additional features of the program

The program runs on personal computers that are "IBM-compatible". The program is written in Modula-2, and so is, as the name suggests, highly modular

[9]. This has made it easy to maintain.

As to be expected, much of the program overhead goes to support the user interface, which implements many of the programming paradigms that are currently standard, such as windows, context-sensitive help, shallow menu structure, and so on. The scheduling capability represents only about 7 out of 10,000 lines of source code.

The program produces a variety of reports and schedules that are important to the functioning of the job fair. Managerial aids include mailing labels, alphabetical lists of participants, and a master schedule that lists all meetings by time, and within each time, alphabetically by name of law firm and again by name of student.

The program also prints individual schedules for each participant. To prepare the individual schedules the program includes an integrated word processor with which the user designs a template for printed schedules. The template gives general information that is needed by all participants, such as the name of the job fair, the location and date, directions on how to get there, and times for individual meetings and for group events. The user can design the template to suit himself. In addition to providing general information, the user embeds tokens in the template to indicate where information particular to an individual should be inserted in the template. For example, the name of the participant will be printed wherever the user types the token `$name`. Similarly, the name of the next person (if any) to meet, and where, will be printed wherever the user types `$whom-to-meet`. A portion of a template might look like Figure 4. Notice that it is the template that translates the abstract times of the computer program to clock times.

Before printing, a "template compiler" checks the template for consistency. For example, the template must include at least a `$name` token as well as a `$whom-to-meet` token for each possible meeting time. The template compiler is transparent to the user unless it finds an error, in which case it tells the user what the inconsistency is and places him instantly back in the word processor to fix it.

Southeastern Public Interest Job Fair

5 and 6 November 1988

`$name`

Saturday, 5 November

08:00–08:30	Check in; coffee and doughnuts
08:30–09:00	`$whom-to-meet`
09:05–09:35	`$whom-to-meet`
09:40–10:10	`$whom-to-meet`
10:10–10:25	Break
10:30–11:00	`$whom-to-meet`
11:05–11:35	`$whom-to-meet`
11:40–12:10	`$whom-to-meet`
12:10–13:10	Lunch

⋮

Figure 4: Portion of a template from which individual schedules can be printed.

Another useful feature is that participants can print messages to each other on their schedules. For example, XYZ Legal Services can remind its interviewees to bring a legal writing sample. Then the following would appear on the schedule of student John Doe, who is to meet XYZ Legal Services during the third time period of the job fair.

09:40–10:10 XYZ Legal Services Room 32 Bring writing sample

7 Managing the job fair

Students generally receive fewer interviews than they requested because the law firms screen their résumés. Therefore it is important that the students receive copies of their schedules in advance so they can decide whether the trip will be worthwhile. (There are generally 3–4 students who receive no interviews.) Since our computer program schedules every mutually-requested meeting, the

job fair is more likely to be worthwhile. Furthermore, by scheduling quickly and conveniently, everyone receives schedules far enough in advance that they can qualify for discounted fares for travel, or, if they decide not to attend, can avoid purchasing non-refundable airline tickets.

Although the job fair has been pleased to give people copies of their schedules in advance, this has caused problems. In particular, each job fair has displayed an alarming tendency to unravel. Naturally enough, each person requires some minimum number of interviews to justify the cost of the trip, and if they receive fewer than this number of interviews they might cancel. This in turn reduces the number of meetings for others and so reduces their incentive to attend the job fair. The danger is that these withdrawals might set off a chain reaction that destroys the job fair in a cascade of cancelled meetings.

Since students request relatively few meetings, their registrations have been particularly volatile. In addition, students have been more apt to cancel appointments for vague or unconvincing reasons. Only once has a law firm cancelled meetings. Fortunately, cancellations have always petered out, with losses of 10–15% of the total meetings originally scheduled.

We have handled cancellations by giving participants revised schedules when they arrive at the job fair. At a central bulletin board, any law firm that has available time can solicit additional interviews. This spot market has been unobtrusive, small, and efficient.

Finally, we admit that, despite all our planning and self-assurance, there was an exquisite suspense at the first job fair, in which we worried whether some hidden miscalculation might have produced conflicting schedules. We are all familiar with the social chaos that quickly envelops a planeload of travellers for whom the seating assignments are in conflict. We winced to imagine that scene magnified several times.

We are happy to report that no conflicts have occurred.

Acknowledgements

We have enjoyed the collaboration of the Atlanta Legal Aid Society and the Georgia State College of Law.

We thank Marian Burge, Steven T. Hackman, and John H. Vande Vate for their comments on an earlier draft.

References

[1] M. W. Carter (1986). "A survey of practical applications of examination timetabling algorithms", *Operations Research* 34(2):193–202.

[2] M. A. H. Dempster (1971). "Two algorithms for the time-table problem", in *Combinatorial Mathematics and Its Applications* (ed. D. J. A. Welsh), Academic Press, New York, pp. 63–85.

[3] H. N. Gabow (1976). "Using Euler partitions to edge color bipartite multigraphs", *International Journal of Computer and Information Sciences*, 5(4):345–354.

[4] H. N. Gabow and O. Kariv (1982). "Algorithms for edge coloring bipartite graphs and multigraphs" *SIAM J. Comput.* 11(1):117–129.

[5] D. Gale and L. S. Shapley (1962). "College admissions and the stability of marriage", *American Mathematical Monthly* 69:9–14.

[6] A. Gibbons (1985). *Algorithmic Graph Theory*, Cambridge University Press.

[7] K. L. McCroan. "Time-tabling", Ph.D. dissertation (in process), Georgia Institute of Technology, School of Industrial and Systems Engineering, Atlanta, GA 30332.

[8] D. de Werra (1970). "On some combinatorial problems arising in scheduling", *INFOR* 8:165–175.

[9] N. Wirth (1985). *Programming in Modula-2 (third, corrected edition)*, Springer-Verlag, New York.

Appendix

While Lemma 4 is sufficient for the purposes of the job fair, it is worth noting that the algorithm can be extended slightly to guarantee an even stronger condition: that if meeting (u, v) is scheduled for time t, then either u or v has consecutive meetings through time t. We call this condition "local consecutivity". A job fair whose meetings satisfy local consecutivity might be imagined more impervious to criticism, since if two people want to reschedule their meeting for an earlier time, they can see for themselves that this must come at the expense of someone else. In contrast, when local consecutivity does not hold, it could happen that the two people are both available at nearby times early in the job fair. This might tempt them to ask that the meeting be rescheduled earlier since it is not locally apparent that this could inconvenience others.

The condition of Lemma 4 is weaker than local consecutivity. Consider, for example, meetings (u, v) and (v, w), where $d_u > d_v > d_w$. It can happen that (u, v) is scheduled for time d_u and (v, w) is scheduled for d_v, in which case neither v nor w would have consecutive meetings through time d_v.

Algorithm RESCHEDULE preserves local consecutivity except possibly for the last person in the rescheduling path. However the algorithm can be revised so that after conflicts have been resolved, it continues rescheduling to enforce local consecutivity. The revised algorithm has the same $O(|E|)$ complexity to schedule a new meeting. See [7] for details.