ON THE REPEATED PARTITION SCHEDULING PROBLEM

Zvi Drezner*

In this paper we present the following scheduling problem. nk people are to be divided into k groups of n people each. Such a division is repeated d times (each time a different partition is scheduled). It is required that each person meets every other person (i.e. is in the same group with him) at least once. The objective is that each person will meet with other participants about the same number of times. The problem is formulated and heuristic algorithms proposed for its solution. Extensive computational experiments are reported.

A meeting organizer suggested the following problem. Twelve persons are participating in a conference for seven days. They are divided into three groups of four persons each. The grouping usually changes from one day to another. It is important that each person meets every other person (i.e. be with him in the same group) at least once. Also, it is desirable that each pair meets about the same number of times as the other pairs. Throughout the paper we refer to this specific problem (with three groups of four people each over seven days) as *the particular problem*. The particular problem originated in planning a golf tournament. For this particular problem there are 66 possible pairs and 126 pair-wise meetings throughout the seven days. This means that on the average each pair meets close to twice (the average is 21/11). Therefore, the best possible solution is having 60 pairs meet twice and six pairs meet once. However, such a solution has not been found yet.

It is relatively simple to find schedules for which each person meets every other person. However, when arrangements are randomly generated, such feasible arrangements occur only once or twice in 10,000 trials. Avoiding four or more meetings can be achieved by trial and

Zvi Drezner is a Professor in the Department of Management Science at California State University, Fullerton.

Manuscript received February, 1996, Revised, May, 1997 & September, 1997

error (the maximum possible number of meetings is seven). The objective is to find a schedule where each pair meets once, twice, or three times, and the number of pairs who meet three times is minimized.

After a long manual effort for some years, the organizer found a solution with twelve pairs meeting three times (and 36 pairs meeting twice, eighteen pairs meeting once). By randomly generating computerized solutions and crude improvement procedures a solution with nine pairs meeting three times was found. The algorithms proposed in this paper yielded a solution with only three pairs meeting three times (and 54 pairs meeting twice, nine pairs meeting once).

The general problem is related to personnel assignment problems [1-5]. Several business applications are described in the next section.

APPLICATIONS

Applications in scheduling and planning of operations are quite common. Such business applications are presented for the particular problem of 12 objects, 3 groups of 4 objects each, and seven days. Applications of the same type may, of course, have different numbers of objects, groups, and days.

1. An operation uses 12 machines (or workers, or any type of resources). There are 21 tasks to be completed in one day. Each task requires 4 machines, so 3 tasks can be performed simultaneously and seven periods are required for the completion of the whole project. The schedule is planned ahead of time so no changes are possible during the day. If one machine is out of order for a task, (or a worker did not show up, or a resource is unavailable), the task can be completed but some extra cost is incurred. However, if two machines are out of order for the same task, the task cannot be completed which adds a significant cost to the operation. We would therefore wish to design a schedule such that the number of times that each pair of machines is assigned to the same task is about the same. If a certain pair of machines is assigned many times to the same task, then the operation will lose an excessive amount if this particular pair of machines happens to be out of order. Assume, for example, that the probability that a machine is out of order is 0.02 (2%). The probability that all machines are operating and the project is completed without any extra cost is 0.98^{12} $=0.7847$. The probability that exactly one machine is not operating is 0.1922. The probability that exactly two machines are out of order is only 0.0216, which leaves a probability of 0.0015 that three or more machines are inoperative. Any schedule will have exactly seven tasks for each machine. Therefore, the extra cost incurred when one machine is out of order is the same for all configurations and we have no control over it. By selecting a good schedule, we can get quite a uniform distribution of losses when two machines are inoperative. The probability that three or more machines are inoperative is so low, that it can be disregarded in the calculations.

2. Twelve workers perform in a very stressful environment. The task requires dividing the workers into three groups of four workers each to perform three different activities. Seven sets of three activities each are required. If two workers are assigned to the same team too many times, they may "get on each other's nerves". The objective is to arrange a schedule such that every pair meets about the same number of times. No pair meets four times and the number of pairs which meet three times is minimal.

3. Twelve drugs are to be tested for possible harmful interactions between pairs of drugs. In order to check all combinations one needs to perform 66 experiments in pairing drugs. This might be too expensive. Three labs are available for such experiments and four different drugs can be administered simultaneously in each experiment. The labs can be booked for seven experiments each. A testing scheme needs to be designed such that in each of the seven days the twelve drugs are divided among the three labs, four drugs to each. It is necessary that each pair of drugs will be tested at least once. Furthermore, it is desirable that every pair of drugs will be in the same group about the same number of times. Such a scheme reduces the number of experiments from 66 to 21.

4. Twelve commercials are prepared for broadcasting. Only four slots are available in a particular program. You wish to test these commercials and find the best combination of four commercials. The effectiveness of each individual commercial depends on the other commercials selected for showing. Some pairs of commercials enhance the effectiveness of one another, and some pairs may reduce the effectiveness. Information about the interactions between various commercials is not available. There are $\binom{12}{4}=495$ possible selections of four commercials out of the twelve available ones. We do not have the resources to check all of these possible selections. Twenty one groups of people are available for testing combinations of four commercials, and there are 3 rooms available for simultaneous showing. We would like to select seven partitions of the twelve commercials into three groups of four each. Our objective is to have each pair of commercials represented in these groups about the same number of times so that these 21 groups will evenly cover the spectrum of possible pairing.

THE GENERAL PROBLEM

Let

P be a partition of people to groups over several days,

n be the number of people in each group,

k be the number of groups,

d be the number of days.

$m_{rs}(P)$ is the number of times pair *(r, s)* meets for a partition *P*,

$N(P,i)$ is the number of pairs that meet *i* times for a partition *P*.

A Partition *P* is termed *feasible* if $N(P,0)=0$. In any partition *P* each person meets *n-1* other people each day for a total of $nk(n-1)/2$ pairwise meetings during the day. The total number of meetings is therefore *dnk(n-1)/2*. The total number of pairs is *nk(nk-1)/2*, and therefore the average number of meetings between pairs is *d(n-1)/(nk-1)*. These calculations can be summarized in the following properties:

Property 1: $\sum_{i=0}^{d} iN(P,i)=\sum_{s=1}^{nk-1}\sum_{r=s+1}^{nk} m_{rs}(P)=\frac{dnk(n-1)}{2}$.

Property 2: $\sum_{i=0}^{d} N(P,i)=\frac{nk(nk-1)}{2}$.

Property 3: The average number of meetings, *A*, is: $A=\frac{\sum_{i=0}^{d} iN(P,i)}{\binom{nk}{2}}=\frac{d(n-1)}{nk-1}$.

The number of possible partitions in one day, *D*, is:

$$D=\frac{(nk)!}{n!^k k!} \qquad (1)$$

and therefore, the total number of possible partitions in all *d* days, *T*, is: $T=\binom{D-1}{d-1}$. For the particular problem *n=4*, *k=3*, and *d=7* yields *D=5,775* and $T\approx 5\cdot 10^{19}$.

We distinguish between "perfect problems" where the average number of meetings (property 3) is integer, and imperfect problems when the average is not an integer. Perfect problems may have solutions for which all pairs meet exactly the same number of times. For example, *k=n* and *d=n+1* yield *A=1*. We found solutions for *n=3,4,5* in which all people meet exactly once. In Table 1 we present such a solution for the *n=3* case.

Table 1: *n=3*, *k=3*, and *d=4*

Day	Group 1	Group 2	Group 3
1	1 2 3	4 5 6	7 8 9
2	1 4 7	2 5 8	3 6 9
3	1 5 9	2 6 7	3 4 8
4	1 6 8	2 4 9	3 5 7

For perfect problems the objective is clear: max $\{N(P,A)\}$ subject to $N(P,0)=0$ by the best selection of P. Define for a number X: $\lfloor X \rfloor$ and $\lceil X \rceil$ as X rounded down or rounded up, respectively. The objective for perfect or imperfect problems is:

$$\max_{P} \{ N(P,\lfloor A \rfloor) + N(P,\lceil A \rceil) \}$$
$$\textit{subject to:} \qquad (2)$$
$$N(P,i)=0 \ \textit{for} \ i \le \max\{0,\lfloor A \rfloor -2\}$$
$$N(P,i)=0 \ \textit{for} \ i \ge \lceil A \rceil +2$$

Note that if $1<A<2$, as in the particular problem, then $N(P,i)>0$ are allowed only for i=1,2, and 3. Therefore, the objective is equivalent to minimizing $N(P,3)$.

AN INTEGER PROGRAMMING FORMULATION

There are D possible partitions in a day by equation (1), and there are $R=nk(nk-1)/2$ possible pairs. Define the matrix $\{a_{ij}\}$ for $i=1,...,D$, and $j=1,...,R$, such that $a_{ij}=0$ if in partition $P=i$ pair j is in two different groups and $a_{ij}=1$ if they are in the same group. For simplicity we assume that the same partition will not be selected more than once during the d days. We define x_i for $i=1,...,D$ as a 0-1 variable. $x_i=1$ if partition i is selected in any of the days, and $x_i=0$ otherwise. (If a partition is allowed for more than one day, then x_i should be defined as an integer betwee zero and d.) The number of times that pair j is meeting is: $\sum_{i=1}^{D} a_{ij}x_i$. As it is proved in Lemma 1

below an equivalent objective function is to minimize $\sum_{j=1}^{R} |\sum_{i=1}^{D} a_{ij}x_j - \frac{\lceil A \rceil + \lfloor A \rfloor}{2}|$.

Therefore, an integer programming formulation is:

$$Min\{\sum_{j=1}^{R} |\sum_{i=1}^{D} a_{ij}x_j - \frac{\lceil A \rceil + \lfloor A \rfloor}{2}|\}$$
$$\textit{subject to:}$$
$$\lfloor A \rfloor -1 \le \sum_{i=1}^{D} a_{ij}x_i \le \lceil A \rceil +1 \ \forall 1 \le j \le R \qquad (3)$$
$$\sum_{i=1}^{D} x_i = d$$
$$x_i \in \{0,1\}$$

Note that the absolute value in the objective function of (3) can be "linearized" in the customary way by defining *2R* variables y_j^- and y_j^+ for $j=1,\dots,R$ and equating quantity j in the absolute value to $y_j^+ - y_j^-$ as a constraint and replacing the absolute value of term j by $y_j^+ + y_j^-$.

Let $I=\lfloor A \rfloor$. I is a given number independent of P.

Lemma 1: Subject to the constraints in equation (3), the objective function of (2) is equivalent to the objective function of (3).

Proof: The objective function of (3), written in terms of $N(P,i)$ is:

$$Min\{\sum_{i=0}^{d} N(P,i)|i-\frac{\lceil A \rceil+\lfloor A \rfloor}{2}|\}.$$

We distinguish between two cases: perfect and imperfect problems. For perfect problems (i.e. $A=\lceil A \rceil=\lfloor A \rfloor$), the constraints entail at most three non-zero $N(P,i)$ for $i=I-1, I, I+1$. Since the sum of the $N(P,i)$'s is constant by Property 2, maximizing $N(P,I)$ is equivalent to minimizing the sum of $N(P,I-1)+N(P,I+1)$. For imperfect problems, a feasible solution may have at most four positive $N(P,i)$'s: $N(P,I-1)$, $N(P,I)$, $N(P,I+1)$, and $N(P,I+2)$. The objective function of (3) to be minimized is: $F=1.5N(P,I-1)+0.5N(P,I)+0.5N(P,I+1)+1.5N(P,I+2)$. By property 2: $F=0.75nk(nk-1)-(N(P,I)+N(P,I+1))$. Therefore, minimizing F is equivalent to maximizing $N(P,I)+N(P,I+1)$. =

Another interesting lemma is that minimizing the sum of squares of the number of meetings is an equivalent objective. Since the mean of the number of meetings is constant by property 3, minimizing the variance of the number of meetings is equivalent to our objective.

Lemma 2: Subject to the constraints of (3), minimizing the sum of squares of the number of meetings $\sum_{s=1}^{nk-1}\sum_{r=s+1}^{nk} m_{rs}^2(P)=\sum_{i=0}^{d} i^2 N(P,i)$ is equivalent to maximizing $N(P,I)+N(P,I+1)$.

Proof: For a given number J:

$$\begin{aligned}\sum_{i=0}^{d} i^2 N(P,i) &= \sum_{i=0}^{d} (i-J)^2 N(P,i) + 2iJN(P,i) - J^2 N(P,i)\\ &= \sum_{i=0}^{d} (i-J)^2 N(P,i) + 2J\frac{dnk(n-1)}{2} - J^2\frac{nk(nk-1)}{2}.\end{aligned} \quad (4)$$

Therefore, minimizing the sum of squares is equivalent to minimizing the sum of squares of the number of meetings adjusted by subtracting a constant. For perfect problems: there are at most three positive *N(P,i)*'s: for *i=I-1, I, I+1*. By applying *J=I* in (4), the objective function is identical to that of (3) because absolute value and a square of -1, 0, and 1 are the same. For imperfect problems apply *J=I+0.5* in (4). There are at most four positive *N(P,i)*'s and after subtracting *J=I+0.5* the sum of squares is equivalent to:

$$\frac{9N(P,I-1)+N(P,I)+N(P,I+1)+9N(P,I+2)}{4}=\frac{9}{4}\frac{nk(nk-1)}{2}-2[N(P,I)+N(P,I+1)]$$

which proves the equivalency between the two objective functions.

HEURISTIC ALGORITHMS

The integer programming formulation lends itself to very large problems even for moderate values of *n*, *k*, and *d*. We therefore explore heuristic algorithms for getting good solutions for the problem. Heuristic algorithms through performing pair exchanges are proposed. A random partition for *d* days is generated and improvements through pair exchanges are considered. Such an approach can be repeated many times and the best solution obtained that way, is selected. In fact, as is demonstrated in the computational results section, these heuristics are very fast for problems of the size of the particular meeting problem.

Three rules for determining pair exchanges were examined. Let *M* and *m* be the maximal and minimal number of meetings for a partition *P*. If *M=m* (for perfect problems) or *M=m+1* (for imperfect problems), then the best possible solution has been found. The first two rules try to lower *N(P,M)* (and alternatively, if this cannot be achieved, lower *N(P,m)* without increasing *N(P,M))*. These rules are quite "natural" for that objective.

Rule 1: Consider all the pairs that meet *M* times. For each pair find the *M* instances where they are assigned to the same group. Check all pair exchanges between one member of the pair and another person in a different group. An exchange is performed if no other pair which now meets *M-1* or *M* times will meet one more time.

Rule 2: For each pair *(r,s)* that meets *m* times check all pair exchanges that move *r* and *s* to the same group (if they were not in the same group before) as long as the following holds: (i) no other pair that met *m* or *m+1* times will meet one fewer times, (ii) no pair will meet *M+1* times, and (iii) the number of pairs who meet *M* times does not increase.

These two rules were designed to ensure no cycling. It is proposed to apply Rule 1 first and if no improvement is possible by Rule 1, apply Rule 2 and if exchanges were made alternate between the rules until no further exchanges are possible by any rule.

Lemma 3: Applying Rule 1 and Rule 2 alternately must terminate in a finite number of iterations.

Proof: All Possible P's can be arranged by the following lexicographic order. Each P has a value for M and m with $\#M$ pairs meeting M times and $\#m$ pairs meeting m times. Compare partitions P_1 and P_2 with M_1, m_1 and M_2, m_2, respectively. P_1 precedes P_2 if: $M_1>M_2$ or $M_1=M_2$ and $\#M_1>\#M_2$. If $M_1=M_2$ and $\#M_1=\#M_2$, then P_1 precedes P_2 if $m_1<m_2$ or $m_1=m_2$ and $\#m_1>\#m_2$. This divides all possible partitions to a finite number of sets where the members of each set are lexicographically equal. It can be easily verified that each application of Rule 1 or Rule 2 results in a partition which is greater lexicographically.

Note that a bound on the number of iterations can be established because the number of possible groups can be easily determined. This bound is polynomial in n, k, and d.

Applying Rules 1 and 2 alternately on the particular problem ($n=4$, $k=3$, $d=7$) required 0.038 seconds of computer time on an IBM 486 compatible per starting solution. The best known solution of 9 pairs meeting once, 54 pairs meeting twice, and 3 pairs meeting three times was obtained only once in 150,000 starting solutions. We therefore considered another rule based on Lemma 2.

Rule 3: Check all pair-wise exchanges between persons in different groups and perform the exchange that yields the smallest sum of squares $\sum_{s=1}^{nk-1}\sum_{r=s+1}^{nk} m_{rs}^2(P)$. Stop when no improvement in the sum of squares is possible.

Calculating the change in the sum of squares:

By exchanging a pair (r,s) between two groups on a particular day, m_{rt} or m_{st} for some $t\neq s$, t may increase or decrease by 1. (For simplicity of notation we omit the partition P from $m_{rt}(P)$.) If m_{rt} increases by 1 than the change in the sum of squares is $2m_{rt}+1$, and if it decreases by 1 the change is $-2m_{rt}+1$. There are $2(n-1)$ increases and $2(n-1)$ decreases for a total of $4(n-1)$ such changes. For each prospective pair exchange (r,s) on a certain day, calculate the sum $S(r,s)$ as follows: when m_{rt} is increased by the exchange, add m_{rt} to $S(r,s)$, and when m_{rt} is decreased subtract m_{rt} from $S(r,s)$. The change in the sum of squares is $2S(r,s)+4(n-1)$ and therefore the pair exchange that yields the smallest $S(r,s)$ is exchanged as long as $S(r,s)<-2(n-1)$. If $\min\{S\}\geq-2(n-1)$ no improvement by Rule 3 is possible. Note that the change in the sum of squares is always even (it can be proven that for a certain problem with given n,k,d, the sum of squares is either even or odd for all configurations depending on whether $dnk(n-1)/2$ is even or odd, respectively). The complexity of one iteration of Rule 3 is as follows. Maintaining the vector of m_{rs}'s is dominated by evaluating all possible pair exchanges. The number of possible pair exchanges is $dn^2k(k-1)/2$, and for each pair the calculation of the change in the sum of squares requires the summation of $4(n-1)$ numbers for a total effort of $o(dn^3k^2)$. The algorithm is polynomial because the number of possible sum of squares is limited by $d^2nk(nk-1)/2$ (a tighter bound can be found) and every iteration reduces this number by at least 2.

This rule was proven to be the most effective for the particular problem. 200,000 partitions were randomly generated using a simple pseudo-random number generator and the various rules were applied on these starting solutions. The best known solution was obtained 5,514 times (or about once in 36 cases). Average run time for applying Rule 3 was 0.092 seconds per starting arrangement. Applying Rules 1 and 2 on the final arrangement of Rule 3 increased the run time to 0.106 seconds. It is interesting to note that applying Rules 1 and 2 on the final solution of Rule 3 did not yield more of these best solutions. This means, of course, that this solution cannot be improved by Rules 1 and 2, but surprisingly no other final arrangement of Rule 3 could be improved by Rules 1 and 2 to yield the best known solution. However, this property does not hold in general. For example, the problem with $k=2$, $n=3$ and $d=5$ is a perfect problem with $A=2$. By equation (1) $D=10$ and consequently the total number of possible partitions is $T=126$. By total enumeration it was found that the optimal solution is $N(P,1)=N(P,2)=N(P,3)=5$. Applying Rule 3 and then Rules 1 and 2 for 10,000 randomly generated starting arrangements yielded the optimal solution in all cases. However, applying Rule 3 only obtained the optimal solution only in 2,705 cases out of 10,000.

ADDITIONAL COMPUTATIONAL EXPERIMENTATIONS

We investigated the distribution of outcomes using Rule 3 only on the particular problem for 10,000 randomly generated starting arrangements. In Table 2 the outcomes in the order of frequency are presented.

Table 2: Applying Rule 3 Only

Number of Meetings					Frequency	Sum of Squares
0	1	2	3	4		
0	13	46	7	0	2708	260
0	12	48	6	0	1888	258
0	14	44	8	0	1866	262
0	11	50	5	0	1281	256
0	15	42	9	0	938	264
0	16	40	10	0	306	266
0	9	54	3	0	298	252
1	10	49	6	0	115	260
1	12	45	8	0	114	264
1	11	47	7	0	106	262
1	13	43	9	0	77	266
1	8	53	4	0	75	256
0	17	38	11	0	65	268
0	13	47	5	1	32	262
0	14	45	6	1	31	264
1	14	41	10	0	30	268
0	15	43	7	1	27	266
0	18	36	12	0	12	270
0	16	41	8	1	11	268
1	15	39	11	0	6	270
0	17	39	9	1	5	270
2	10	46	8	0	3	266
2	11	44	9	0	2	268
1	16	37	12	0	1	272
2	12	42	10	0	1	270
2	8	50	6	0	1	262
0	18	37	10	1	1	272

Note that the solution (0,10,52,4,0) was never obtained. It is possible that such solutions do exist but Rule 3 improves them to (0,9,54,3,0). An example of a best known solution for the particular problem is given in Table 3. Nine pairs meet once, 54 pairs meet twice, and only three pairs (3-10, 5-12, and 7-8) meet three times.

Table 3: Arrangement of Meetings for the Particular Problem

Day	Group 1	Group 2	Group 3
1	1 2 3 4	5 6 7 8	9 10 11 12
2	2 4 5 9	1 6 10 12	3 7 8 11
3	3 4 6 10	2 5 11 12	1 7 8 9
4	2 3 8 10	4 7 9 12	1 5 6 11
5	1 3 5 12	4 7 10 11	2 6 8 9
6	2 6 7 12	1 4 8 11	3 5 9 10
7	4 5 8 12	1 2 7 10	3 6 9 11

We experimented with a larger perfect problem of *k=4*, *n=3*, and *d=11* for which *A=2*. No perfect solution with all 66 pairs meeting twice was obtained. The best solution of 2 pairs meeting once, 62 pairs meeting twice and 2 pairs meeting three times was obtained 77 times starting with 100,000 random solutions. In all 77 cases the same best solution was obtained by Rule 3 alone. Run time was 0.202 seconds per iteration for applying Rule 3 only, and 0.214 seconds when Rules 1 and 2 were applied on the solution of Rule 3. A best known solution is given in Table 4. Pairs 4-8 and 5-10 meet once, and pairs 5-8 and 4-10 meet three times. All other pairs meet twice.

Table 4: Best known Solution for the *k=4*, *n=3*, and *d=11* Problem

Day	Group 1	Group 2	Group 3	Group 4
1	1 2 3	4 5 6	7 8 9	10 11 12
2	4 9 10	1 5 11	3 8 12	2 6 7
3	1 9 11	4 7 12	3 6 10	2 5 8
4	2 7 12	3 10 11	1 6 9	4 5 8
5	2 4 9	3 5 8	1 7 10	6 11 12
6	2 10 12	1 3 4	5 7 11	6 8 9
7	2 4 11	3 7 9	1 8 10	5 6 12
8	2 3 6	5 9 10	7 8 11	1 4 12
9	3 5 7	2 9 11	1 8 12	4 6 10
10	4 7 10	1 2 5	6 8 11	3 9 12
11	2 8 10	5 9 12	1 6 7	3 4 11

CONCLUSIONS

The problem of repeated scheduling of a group of objects into smaller groups is considered. A group of *nk* objects is to be divided into *k* groups of *n* members each. Such a division need to be repeated *d* times. Our objective is that each pair of objects meets (i.e., it is in the same group) at least once. Also, we would like each pair of objects to meet about the same number of times.

Applications can be found in tournament scheduling, personnel assignment, production scheduling, advertising, experimental design, and others.

Several algorithms for the solution of this problem are suggested and tested. These algorithms can be applied to reasonable size problems. Large problems can employ metaheuristic approaches such as tabu search, simulated annealing, or genetic algorithms. The examination of such procedures is left for future research.

ACKNOWLEDGEMENT:

Part of this research was done while the author was on sabbatical leave at the Hong Kong University of Science and Technology, Kowloon, Hong Kong. The author would like to thank Professor H. Rutemiller of California State University-Fullerton for suggesting the problem

REFERENCES

Mehrez, A. and D. Shinhar. (1982). "Organizational Optimization in Personnel Placement,"***Omega*****,** 10, 331-333.

Mehrez, A. Y. Yuan, and A. Gafni. (1988). "Stable Solutions vs. Multiplicative Utility Solutions for the Assignment Problem," ***Operations Research Letters***, 7, 131-139.

Roth A. (1984). "Stability and Polarization of Interests in Job Matching," ***Econometrica***, 52, 47-57.

Roth A. (1985). "Conflict and Coincidence of Interest in Job Matching: Some New Results and Open Questions," ***Mathematics of Operations Research***, 10, 379-389.

Yufei Y., A. Mehrez, and A. Gafni. (1992). "Reducing Bias in a Personnel Assignment Process Via Multiplicative Utility Solution," ***Management Science***, 38, 227-239.