

*Working Paper Series*

HEARIN CENTER  
FOR  
ENTERPRISE SCIENCE

HCES-04-04

Guided Design Search in the Interval-bounded  
Sailor Assignment Problem

By

Mark W. Lewis  
Karen R. Lewis  
Barbara J. White



*The University of Mississippi*

Director, Keith Womer  
School of Business Administration  
The University of Mississippi  
Post Office Box 1848  
University, MS 38677-1848  
(662) 915-5820  
<http://hces.bus.olemiss.edu>

---

# Guided Design Search in the Interval-Bounded Sailor Assignment Problem

Mark W. Lewis<sup>a\*</sup>, Karen R. Lewis<sup>a</sup>, Barbara J. White<sup>a</sup>

a *School of Business, University of Mississippi University, Mississippi 38677, USA*  
*mlewis, klewis, bwhite@bus.olemiss.edu*

---

**Abstract** —Guided Design Search is a preprocessing technique that uses experimental design sampling techniques to efficiently generate the estimated effects of binary decision variables on objective function values. The interval-bounded sailor assignment problem is a new model for optimizing the ships manned with teams of sailors with specific skills. Interval bounding requires certain activities to be either at a 0 level or at some minimum level – this is in contrast to requiring a minimum level or allowing very small levels, e.g. training classes with less than 5 students are not offered. Using Guided Design Search to preprocess the interval-bounded sailor assignment problem, we show that setting a very small percent of the total number of decision variables greatly improves solution quality and time to optimality.

---

Keywords: Sailor assignment, Interval bounds, Elastic constraints, Experimental design, Guided Design Search

---

\* This research was performed under ONR Grant N000140310621

## 1. Introduction

For years, there has been research devoted to the problem of assigning sailors to various Navy positions, known as jobs or billets. There are currently 500,000 sailors in about 100 different job types. One of the problems in assigning sailors is that there are generally more billets than sailors. Therefore, not all jobs are filled. Hence it is advantageous to allocate sailors carefully and provide proper balance. This is often paraphrased as “The right sailor in the right job at the right time”.

In this paper, we present a new interval bounded network flow model for this problem, which includes the majority of the supply chain: sailors, training, job / skill categories, and ships. Interval bounding variables are used to decide which teams of sailors and skill categories are assigned to which ships in order for that ship to be operational. They are also used to decide which training classes should be offered. The supply chain aspects of the model can be easily extended to include activities related to sailor recruiting (i.e. more resources in the supply chain) and to combining ships to create fleets.

Modeling using interval bounds can create difficult mixed-integer linear programs (MIPs) mainly because the LP relaxation typically does not provide a good lower bound for solvers such as CPLEX. Guided Design Search (GDS) is a solution technique with various uses, for example, it can be used to quantify the average effects of a decision variable; guide a search of the solution space; and/or reduce the solution space by setting certain important decision variables to their most beneficial value. In this paper, we supplement CPLEX with GDS to identify important variables prior to starting the standard branch-and-bound search. In our testing we found GDS was over 10x faster on problems solved to optimality and provided better solutions on larger problem instances not solved to optimality. In some cases, GDS was able to deploy several ships, while CPLEX alone was unable to find any feasible solution.

The paper is outlined as follows: a description of the Navy’s sailor assignment problem, the mathematical description of the new model employed, a description of Guided Design Search, computational results, and summary and conclusion.

## 2. Description of the Sailor Assignment Problem

In the current process, a *detailer*, while interacting with the sailors to be assigned, assigns an individual sailor to a billet manually. The general structure of the sailor assignment problem is described as follows. In order to be assigned a job, sailors have to meet certain skill requirements. A sailor may have up to 5 skills listed in his/her record. A job may require one or two skills. A sailor may, therefore, already have the skill required or may need to attend a training class to obtain the skill before starting the new job. It is preferable to choose a sailor for a billet who already possesses the necessary skills, for several reasons. First, if the sailor has a skill, it is better to utilize that skill than to train another sailor. Second, there are costs associated with the training class itself and with travel to and from the class. Also, the time spent in the training class pulls the sailor away from other duties. Training classes are not generally held if only one or two sailors are required to be trained. It is much more efficient to train a certain number of sailors at one time. This is an example of interval bounding. A benefit of training is that the sailor is typically qualified for more than one job.

The detailer cannot be expected to consider how one assignment will impact the ability to optimize the entire set of assignments. For many years, the Navy has tried to automate this process. Most researchers have tried to solve this problem with some variation of the assignment problem. Klingman and Phillips [84] developed a network

model with a multi-criteria objective to address the assignment problem for the Marine Corps, since there are various policies and goals that the Navy is trying to meet in making assignments. Liang and Thompson [87] proposed modeling the problem as a transportation model with a side constraint. The side constraint represents the Navy's allocation goals. It controls the way in which people are allocated to various groups, such as region or type of duty. Liang and Buclatin [88] and Ali, Kennington, and Liang [93] modeled the problem as a network model with side constraints to consider training resource utilization. These models use side constraints to limit the number of seats in a class.

These approaches assign sailors simultaneously to jobs to optimize the set of assignments. Holder [04] uses a different approach to the problem. Since detailers give the sailor choices in assignment, his approach is to solve the assignment problem in a way which reveals any assignment which can be made in some optimum. These assignments can then be used to select a choice of jobs for each sailor. His method uses a simple cost function which can create multiple sailor assignments with the same cost. Then he solves the problem using a path-following interior-point algorithm. Any sailor-job pair with a nonzero value in the optimal solution is an optimal assignment in some basic optimal solution to the problem.

In our model we have sailors who are to be assigned to various job types, which is similar to the previous models. We don't look at these jobs independently, however, but take into account how the various billets combine on a ship. We assume that a ship cannot be deployed unless a minimum number of job types are satisfied. In addition, for each of these job types, a minimum number of billets must be filled. Naturally, there is also a maximum capacity in each job type and for the number of job types.

Our model, therefore, must address skills, training, job categories, and ships. There are various training classes to accommodate other matches, but there is a minimum/maximum number for each training class, so that the fixed cost of a class will not be incurred for too few students. In our model, sailors are trained, if necessary, and then assigned to a job category. The model also takes into account requirements for teams of sailors versus individual sailors. Figure 1 illustrates the basic network flows, although for simplicity the interval bounding constraints are not shown.

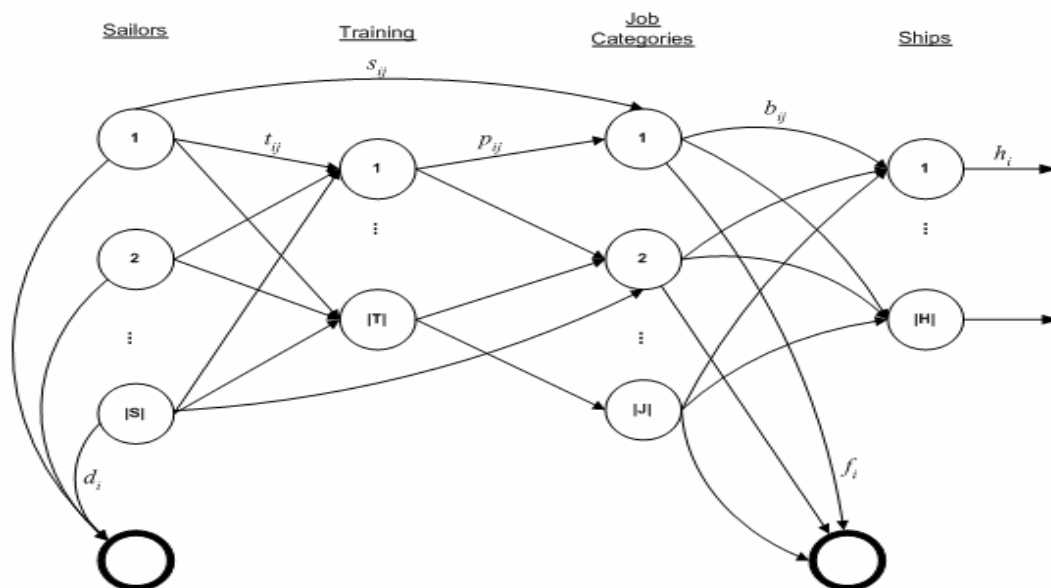


Figure 1. Network Flow Diagram for Sailor Assignment

## 2.1 Derivation of an objective function

The objective function of the interval-bounded network flow model has multiple goals. For example, the main costs to be minimized are the Permanent Change of Station (PCS) costs, training costs, and the penalties for not assigning sailors in this time window (or assigning them to shore duty). Associated with costs are benefits. These are negative costs and include the benefit of meeting training class levels, or of filling a ship with the proper number of teams of sailors having the right skills. Although all costs may be difficult to quantify, the relative magnitude of the costs and benefits are intuitive, i.e. the relative benefit of having a ship operational is orders of magnitude more important than the PCS cost of an individual sailor.

Although it may be difficult in practice to achieve consensus on the exact form of the objective function and its associated costs, quantification is necessary in gaining an understanding of the relative importance of the various components of the problem. GDS can aid in this understanding by providing estimated effects of individual decisions, allowing them to be ranked by the type of decision being made. For example, if the average effect of assigning a sailor a particular job is much higher than any of his other assignments, then perhaps that sailor should be motivated to take that job. The point being that the original objective function coefficients can be modified or eliminated after GDS quantifies their effects.

Another difficulty when using goal programming is various factions do not feel adequately represented by their quantifications. For example, what is the relative importance of staying under the PCS budget versus the training budget? The model presented in this paper does not include specific budget constraints, and although it could include them, hard budget constraints are not necessarily realistic. The model does provide for minimization of specific budgets within the context of the other objective function variables. For example, the relative benefit of deploying a ship at the required job skill / team levels is much greater than the benefit of holding a training class.

## 3. Mathematical Model

The objective function variables that reduce cost are those that represent the benefit of reaching a minimum level of some commodity. The three types of binary variables used for interval bounding represent whether or not a ship is deployed, whether or not there are enough sailors from a job category for a ship, and whether or not there are enough students to attend a course. The objective function variables that add costs and are to be minimized are: PCS costs, training costs, shore duty costs, and the cost of not being able to assign a sailor at this time. Note that the interaction effect between deploying two or more ships is not specifically modeled, but could be added to the model. The interval bounded model assigning sailors to skilled teams, which are then assigned to ships more accurately models the Navy personnel supply chain. Following are the representations for each set and the sets they represent.

H	All ships
J	All job openings
$J^R_j$	All jobs for which training class $j$ qualifies a sailor
$J^S_i$	All jobs for which sailor $i$ is qualified
S	All sailors

The binary variables used in this model are defined in the following table.

$h_i$	Represents whether or not ship $i$ meets the required level of each of the job categories required for deployment
$m_{ij}$	Represents whether or not the required number of sailors from job category $i$ needed for ship $j$ is met
$r_i$	Represents whether or not enough students enroll for training class $i$ to be held
$s_{ij}$	Represents whether or not sailor $i$ is directly assigned to job $j$
$t_{ij}$	Represents whether or not sailor $i$ is assigned to training class $j$

The nonnegative variables used in this model are defined in the following table.

$b_{ij}$	Number of sailors assigned to job $i$ on ship $j$
$d_i$	Represents whether sailor $i$ is assigned to a job or training or is not assigned to anything
$f_i$	Number of sailors assigned to job category $i$ that are not assigned to any ship
$p_{ij}$	Number of sailors assigned to job $j$ after being trained in training class $i$

Parameters are given in the following table.

$c^d_i$	Cost of not assigning sailor $i$
$c^f_i$	Cost of not assigning sailors from job category $i$ to any ship
$c^h_i$	Bonus for having ship $i$ deployed
$c^{p_{ij}}$	Cost of moving sailors from training $i$ to job $j$
$c^r_i$	Bonus for holding training class $i$
$c^{s_{ij}}$	Cost of moving sailor $i$ to job $j$
$c^{t_{ij}}$	Cost of moving sailor $i$ to training $j$
$h^l_i$	Lower limit on the number of job categories needed for ship $i$
$h^u_i$	Upper limit on the number of job categories needed for ship $i$
$m^l_{ij}$	Lower limit on the number of sailors from job category $i$ needed for ship $j$
$m^u_{ij}$	Upper limit on the number of sailors from job category $i$ needed for ship $j$
$r^l_i$	Lower limit on the number of sailors needed for training class $i$
$r^u_i$	Upper limit on the number of sailors needed for training class $i$

A sailor can be assigned directly to a job or to training and then a job. A single training class may train a sailor for one job or multiple jobs. The variables  $d_i$  and  $f_i$  are dummy variables needed to insure model feasibility, since it may not be possible to assign all sailors to jobs and ships. Variable  $d_i$  is used as a binary variable, but does not need to be defined as one because of the way in which it is constrained in the model.

The interval bounded sailor model is defined as:

$$S_1 = \{ \text{minimize} \\ \sum_{i \in \text{Sailors}} c^d_i d_i + \sum_{i \in \text{Jobs}} c^f_i f_i + \sum_{i \in \text{Training}} \sum_{j \in J_i^R} c^{p_{ij}} p_{ij} + \sum_{i \in \text{Sailors}} \sum_{j \in J_i^S} c^{s_{ij}} s_{ij} + \sum_{i \in \text{Sailors}} \sum_{i \in \text{Training}} c^{t_{ij}} t_{ij} + \sum_{i \in \text{Ships}} c^h_i h_i + \sum_{i \in \text{Training}} c^r_i r_i \quad (1)$$

subject to

$$\sum_{j \in J_i^S} s_{ij} + \sum_{j \in \text{Training}} t_{ij} + d_i = 1 \quad \forall i \in S \quad (2)$$

$$\sum_{i \in \text{Sailors}} t_{ij} > r^l_j r_j \quad \forall j \in T \quad (3)$$

$$\sum_{i \in \text{Sailors}} t_{ij} < r^{uj} r_j \quad \forall j \in T \quad (4)$$

$$\sum_{j \in \text{Sailors}} t_{ji} = \sum_{j \in J_i^R} p_{ij} \quad \forall i \in T \quad (5)$$

$$\sum_{\substack{i \in \text{Sailors} \\ \ni j \in J_i^R}} s_{ij} + \sum_{\substack{i \in \text{Training} \\ \ni j \in J_i^R}} p_{ij} = \sum_{i \in \text{Ships}} b_{ji} + f_j \quad \forall j \in J \quad (6)$$

$$b_{ij} > m^{l_{ij}} m_{ij} \quad \forall i \in J, \forall j \in H \quad (7)$$

$$b_{ij} < m^{u_{ij}} m_{ij} \quad \forall i \in J, \forall j \in H \quad (8)$$

$$\sum_{j \in \text{Jobs}} m_{ji} > h^l_i h_i \quad \forall i \in H \quad (9)$$

$$\sum_{j \in \text{Jobs}} m_{ji} < h^u_i h_i \quad \forall i \in H \quad (10)$$

$$\left. \begin{aligned} s_{ij} &\in \{0,1\}, p_{ij} > 0 && \forall i \in S, \forall j \in J \\ t_{ij} &\in \{0,1\} && \forall i \in S, \forall j \in T \\ h_i &\in \{0,1\} && \forall i \in H \\ m_{ij} &\in \{0,1\}, b_{ij} > 0 && \forall i \in J, \forall j \in H \\ r_i &\in \{0,1\} && \forall i \in T \\ f_i &> 0 && \forall i \in J \\ d_i &> 0 && \forall i \in S \end{aligned} \right\}$$

Constraint (2) requires that every sailor be assigned a job or to training for a job, or if the supply chain cannot handle this sailor, then he is not assigned. Constraints (3) and (4) are interval bounding constraints requiring that if training class  $j$  is held, then the lower and upper bound on sailors must be met. Constraint (5) is a flow balance constraint requiring that the number of sailors entering training equals the number assigned to jobs for which the training qualifies them (the possibility of drop outs are not included in this model). Constraint (6) is the flow balance equation requiring that the number of sailors assigned to a job skill category equals the number assigned from that category to ship or shore duty. Constraints (7) and (8) require that a team of sailors meets specified lower and upper bounds for job categories on ships. Constraints (9) and (10) require that a specified number of sailor teams is needed in order for a ship to be deployed.

A solution will provide which sailor is assigned to which training or job skill, but not exactly which ship or team he may be on or from training exactly which job category he may be assigned to. This is left to the detailer. Thus the solution provided by including all possible options at once is used as a baseline for the detailers to work from when working with individual sailors and for other management personnel to consider as a benchmark to attain. The average effects of decision variables from the GDS preprocessing can also be used to help guide the detailers in their decisions.

#### 4. Guided Design Search Algorithm

Taguchi methods (1989) stress quality control early in the design process in order to enhance quality during production operations. A key element of Taguchi methods is the use of experimental designs. Design of experiments (DOE) is used to help determine the critical factors affecting a given performance parameter, e.g. what temperature and pressure settings optimize product yield. Benefits of DOE testing include unbiased sampling (versus random sampling or simply averaging observations

made during branch-and-bound or a metaheuristic search) and interaction effects estimation (DeVor et al. 1992). *Main effect estimation* is the estimated average effect of a variable on the output parameter which is, in our case, the objective function value.

In a production environment test runs are generally expensive. However, GDS test runs are generally not expensive, with each test run sample usually being solved in fractions of a second. Although testing is fast, the number of test runs per factor grows at an exponential rate. Therefore confounding techniques (also known as fractional experimental designs) are employed to reduce the number of tests for a given number of variables. Variables that are confounded in a fractional DOE have their own effects combined (i.e. confounded) with high-level interaction effects into one estimate (DeVor et al. 1992). Fortunately, sequential testing (e.g. using the “mirror” of the original experiment) can remove the effects of confounding and reveal the main effects (Mason et al. 1989).

Appendix A describes the procedure for generating variable effect estimates via experimental design testing. In summary, an experimental design (DOE) is created which is based on the number of binary variables. For example, problems with 435 binary variables would create a DOE having 512 test runs. The DOE is a table consisting of settings for all binary variables for each test run. For each test run, the binary variables are set at the appropriate values according to the DOE. Typically, with the difficult binary variables set, the problem is quickly solved. For each test run the objective function value is recorded. When all runs in the table have been completed, all test runs are performed using the *mirror* of the original DOE values, in other words, all 0s and 1s are switched. This sequential mirror test allows the confounding elements to be canceled and the main effects on the objective function revealed.

A variable selection rule is used to decide which variables are to be set high or low. This is analogous to identifying outliers in a distribution. The rule we used a priori for all testing was to set the top 10% of all variables whose value was beyond one standard deviation from the entire distribution of sampled variables. On average, a small number of variables were set, on the order of 3 or 4. An example of a typical distribution is shown in Figure 2, including the one standard deviation limit used for outlier identification. In this example, 7 variables are possible outliers, but only the top 10% of these seven would actually be set to 1. Setting too many variables can create infeasibilities and our hypothesis is that a small, select number of variables will have a large effect. Appendix B provides a small example of how DOE testing accomplishes the quantification of variables.

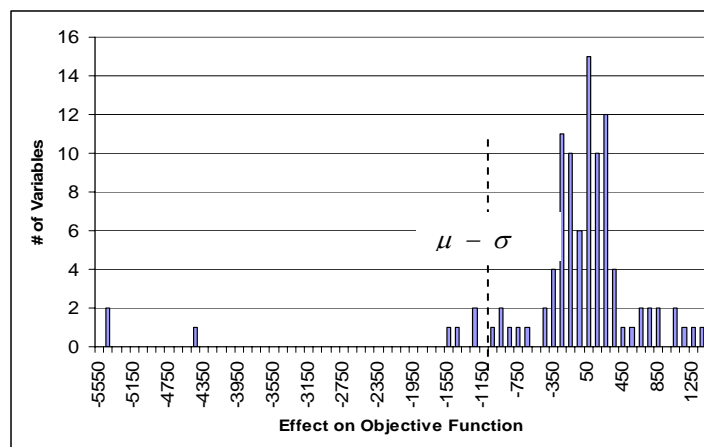


Figure 2. Typical Distribution of Variable Effects on Objective Function

## 5. Computational Results

There are a number of parameters affecting the time to solve and solution quality of problems generated using the model. Since this is a new model, we underwent initial testing to understand some of the sensitivities of problem complexity. During testing, we found that the number of training classes and their characteristics (class size, number of jobs the class qualifies a sailor for, cost, min/max levels) affects problem complexity. Other factors that affect problem complexity are the number of jobs a sailor is directly qualified for, since it affects the amount of training required, the cost associated with assigning sailors shore duty, and the number of jobs to be filled.

To gauge the effect of these Navy parameters, we performed computational testing based on a full factorial experimental design with 4 parameters comparing default CPLEX to GDS/CPLEX. Table 1 shows the problem parameters and total times taken by CPLEX and GDS. The number of decision variables is affected by the number of jobs to fill and the number of jobs a sailor is directly qualified for, i.e. the more jobs to fill, the more possible jobs a sailor can be assigned to, and the fewer jobs he is qualified for, more possible training decisions can be made.

Table 1. Problem Parameters (All problems solved to optimality)

Jobs to Fill	Qualified Jobs per Sailor	Minimum Training class size	Penalty for shore duty	Number of decision variables	CPLEX time (sec)	GDS total time (sec)
500	1	5	10	3946	1779	217
500	2	5	10	3936	7053	771
500	1	10	10	3946	3306	368
500	2	10	10	3936	347	141
500	1	5	500	3946	2995	196
500	2	5	500	3936	812	135
500	1	10	500	3946	3738	180
500	2	10	500	3936	1105	181
1000	1	5	10	3960	1754	222
1000	2	5	10	3946	1013	126
1000	1	10	10	3960	3852	175
1000	2	10	10	3946	411	138
1000	1	5	500	3960	3399	187
1000	2	5	500	3946	1045	149
1000	1	10	500	3960	4143	191
1000	2	10	500	3946	1296	234

The averages for the 16 tests are shown in Table 2. GDS improved the speed of solution by an average of 11 times over CPLEX. Both CPLEX and GDS solved all problems to optimality. Table 2 shows that when sailors were qualified for only 1 billet, CPLEX took much longer than when they were qualified for 2, while GDS took about the same amount of time. On average, it appears that the more direct assignments that can be made, the easier the problem. When there was a larger penalty for assigning sailors to shore duty, GDS solved these problems relatively faster than average, i.e., GDS had slightly less difficulty assigning sailors to ships than CPLEX did. The table also shows that the GDS preprocessing time accounts for about half of the total processing time. Thus, time spent in preprocessing is rewarded during the solution search.

Table 2. Multi-dimensional Summary of CPLEX and GDS

	Average Time (seconds)			Speed-Up (CPLEX / GDS)
	CPLEX	GDS Pre- processing	GDS Total	
Total number of billets to fill				
500	2642	118	274	10
1000	2114	121	178	12
Number of billets a sailor is qualified for without training				
1	3121	116	217	15
2	1635	122	234	6
Minimum number of sailors in a training class				
5	2481	118	250	10
10	2275	121	201	11
Penalty for assigning sailors to shore duty				
10	2439	116	270	9
500	2317	122	182	13

Table 3 summarizes seven problems that were found to be difficult. These problems ran on average for 5 hours before memory limitations forced them to terminate. The GDS solution was the same or better than CPLEX for these difficult problems. The GDS optimality gap was much better than the CPLEX optimality gap. GDS solved two of the problems (2 and 4) to optimality, while CPLEX terminated them with optimality gaps of 36% and 39%. GDS found much better solutions for problems 1, 5, and 6, with CPLEX terminating the latter two with no sailors assigned.

Table 3. Larger Problem Size Summary for CPLEX and GDS

Problem ID	number decision variables	CPLEX time (sec)	GDS time (sec)	CPLEX solution	GDS solution	CPLEX opt. gap	GDS opt. gap
1	7440	19575	48116	-30108	-42686	88%	33%
2	7430	26769	607	-19684	-19684	36%	0%
3	3940	8651	33078	-20048	-20053	39%	35%
4	7430	28870	3149	-19957	-19957	39%	0%
5	7440	18394	30469	7062	-26253	929%	123%
6	7440	15274	12596	7062	-35733	929%	64%
7	3960	3200	6216	-19894	-20053	49%	47%

## 6. Summary and Conclusion

We have found that GDS used in conjunction with the CPLEX solver is an appropriate tool for this new sailor assignment model, finding solutions allowing ships to be deployed when default CPLEX may not. The new sailor model with interval bounds is a more realistic, but more difficult model to solve. GDS preprocessing was successful at identifying important variables and setting a small number of these important decision variables had a dramatic effect on solution time and quality. Future research may be used to explore additional sailor supply chain aspects, including adding recruiting supply nodes and options for combining ships into fleets of ships. Future GDS research may be directed towards grouping the decision variable

quantifications and applying rules by group. For example, it is obvious that the decision to deploy a ship carries more weight than which training class a sailor should take. Therefore, instead of using one rule for all variables, they could be broken into groups and apply different rules for each group.

## References

Ahuja, R.K., T.L. Magnanti, J.B. Orlin, 1993. *Network Flows – Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ.

Barr, R.S., B.L. Golden, J.P. Kelly, M.G.C. Resende, and W.R. Stewart, 1995. Designing and Reporting On Computational Experiments with Heuristic Methods. *Journal of Heuristics* **1** 19-32.

Berger, P.D. and R.E. Maurer, 2002. *Experimental Design with applications in management, engineering, and the sciences*. Duxbury – Thompson Learning.

Box, G.E.P. and R.D. Meyer, 1986. An Analysis For Unreplicated Fractional Factorials, *Technometrics* **28** 11-18.

Chinneck, J. W. and E. W. Dravnieks, 1991. Locating Minimal Infeasible Constraint Sets In Linear Programs. *ORSA Journal on Computing* **3** 2 157-168.

DeVor, R., T. Chang, and J. Sutherland, *Statistical Quality Design and Control: Contemporary Concepts and Methods*. New York, NY: MacMillan Publishing Company, 1992.

Holder, A., 2000. Designing Radiotherapy Plans with Elastic Constraints and Interior Point Methods. *Health Care and Management Science* **6** 1 5 - 16.

Holder, A. Navy Personnel Planning and the Optimal Partition. 2002, Trinity University, Mathematics Technical Report #63, San Antonio, TX, to appear in *Operations Research*

ILOG Optimization Suite. 2002. ILOG S. A., 9 Rue de Verdun, 94253 Gentilly Cedex, France, and ILOG, Inc., 1080 Linda Vista Ave., Mountain View, California 94043, USA.

Karger, D.R., 1999. Random Sampling In Cut, Flow, And Network Design Problems. *Mathematics of Operations Research* **24** 2 383-413.

Kennington, J., 1978. A Survey of Multicommodity Network Flows. *Operations Research* **26** 209-236.

Lewis, M.W., 2004. Guided Design Search in Joint Capacity Allocation Network Problems. University of Mississippi, *Hearin Center Technical Report* **HCES-03-04**.

Linderoth, J.T. and M.W.P. Savelsbergh, 1999. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing* **11** 2 173-187.

Magnanti, T.L., R.T. Wong, 1986. Network Design And Transportation Planning: Models And Algorithms. *Transportation Science* **18** 1-55.

Mason, R., R. Gunst, and J. Hess, 1989. *Statistical Design and Analysis of Experiments*. New York, NY: John Wiley and Sons.

Montgomery, D.C. 1984. *Design and Analysis of Experiments*. New York, NY: John Wiley and Sons.

Rardin, R. and U. Reha, 2001. Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial. *Journal of Heuristics* 7 261-304.

Taguchi, G., A. Elsayed, and T. Hsiang, 1989. *Quality Engineering in Production Systems*. New York: McGraw-Hill.

## Appendix A

To clarify what is meant by GDS, the pseudo-code for the process is provided below. Note that the only problem-specific portions of the algorithm are gauging the size of the penalty to be used with the elastic constraints. Details of the calculations required in experimental designs can be found in (Mason et al. 1989, DeVor et al. 1992, Berger and Mauer 2002, and other introductory and advanced textbooks).

- (1) Create the problem instance using the appropriate elastic model;
- (2) Let  $\mathbf{b}$  be the binary variables sequenced according to the order that they appear in the problem (or by the columns they are assigned) and let the number of binary variables  $b = |\mathbf{b}|$ . Thus,  $b_3$  is the third binary variable.
- (3) Generate the corresponding fractional factorial DOE table,  $T$ , for  $b$  binary variables.  $T$  is an  $n$  by  $m$  zero-one matrix where  $|n|$  is the number of tests needed for  $b$  variables and column  $j$  is assigned (or corresponds) to the binary variable  $b_j$ . Therefore,  $T_{ij}$  specifies at what level variable  $b_j \in \{0,1\}$  must be set for test run  $i$ . The upper bound for the total number of variables that can be tested in a fractional DOE of size  $k$  is  $2^k - 1$ . While the variable can take on the values of 0 or 1, the variable has two levels, low and high. A variable level of low corresponds to -1 and a high level corresponds to 1. Some variables are confounded with interactions of the main effect variables. These variable levels are determined by multiplying the numeric values of the levels of the variables in the interaction.
- (4) Let  $T^M$  be the mirror image of  $T$ .  $T^M$  is created by simply swapping the values of 0 refer to various DOE references for the following note. Certain interaction effect columns in  $T^M$  are now the negative of their value as computed by the corresponding main effect variables. Calculate these column values from the corresponding main effects. Store the results as vector  $\mathbf{g}$  with length  $b$ . The  $i^{\text{th}}$  element of  $\mathbf{g}$  is the sign that corresponds to the  $i^{\text{th}}$  element of  $\mathbf{b}$  in  $T^M$ . The sign will determine whether a test result is added or subtracted during the calculation of variable main effects in step (7).)
- (5) For each test run (row)  $i$  in  $T$ :
  - a) Assign values to each of the  $\mathbf{b}_j$  variables according to row  $i$  and column  $j$  in table  $T_{ij}$ ;
  - b) Solve the problem and record the value of the objective function;
  - c) If the objective value is better than the best, then save the associated basis as a possible starting basis in step (9);
- (6) Perform the mirror test to remove 2-factor confounding. For each test run  $i$  in  $T^M$ :
  - a) Assign values to each of the  $\mathbf{b}_j$  variables according to row  $i$  and column  $j$  in table  $T_{ij}^M$ ;

- b) Solve the problem and record the value of the objective function;
  - c) If the objective value is better than the best, then save the associated basis as a possible starting basis in step (9);
- (7) Calculate the estimate of the average effect each variable had on the objective function value using the results from (4), (5) and (6). See various DOE textbook references for more information.
- (8) Assign priorities and branching direction to every binary variable  $b_j$  based on their estimated average effect on the objective function;
- (9) Integrate the above information into your search technique and solve;

### Appendix B

A small DOE example with 7 possible decision variables  $y_1$ –  $y_7$  is illustrated below. With seven variables, a DOE with eight test runs is created. Four of the seven binary decision variable effects are confounded with the interaction effects of the first three variables. This is called a 2-level fractional factorial with confounding DOE. For each test run, the  $y$  variables are set according to the values in the table, the corresponding LP is optimized and the objective function value is recorded as shown in the table.

While the variable can take on the values of 0 or 1, the variable has two levels, low and high. A variable level of low corresponds to -1 and a high level corresponds to 1. When the testing is complete, the estimate of the average effect is calculated by multiplying the numeric variable level by the objective function value. For example, the estimate of the average effect of setting  $y_1$  from 0 to 1 is  $\sigma(y_1) = (-120 + 101 - 23 + 65 - 98 + 115 - 109 + 88) / 4$ . The calculated values in the table below have been rounded. If the average values in the table below were used in an exact branch-and-bound, then the binary variables would be given corresponding priorities and branching directions, e.g. variable  $y_2$  would be given the highest priority (for variable selection) and its branching direction preference would be to 1, whereas variable  $y_3$  would have lower priority and a preferred direction to 0. As a heuristic, the variable  $y_2$  could be set to 1, simplifying the search process.

Test Run #	$y_1$	$y_2$	$y_3$	$y_1 y_2 = y_4$	$y_1 y_3 = y_5$	$y_2 y_3 = y_6$	$y_1 y_2 y_3 = y_7$	Objective Function Value
1	0	0	0	1	1	1	0	120
2	1	0	0	0	0	1	1	101
3	0	1	0	0	1	0	1	23
4	1	1	0	1	0	0	0	65
5	0	0	1	1	0	0	1	98
6	1	0	1	0	1	0	0	115
7	0	1	1	0	0	1	0	109
8	1	1	1	1	1	1	1	88
Avg.	5	-37	25	6	-7	29	-25	