ORSA Journal on Computing
Vol 5, No 4, Fall 1993

0899-1499/93/0504-0395 $01 25

Network Reoptimization Algorithms: A Statistically Designed Comparison

MOHAMMAD M AMINI / Department of Management Information Systems and Decision Sciences, Memphis State University, Memphis, TN 38157, Email aminim@memstvx1 bitnet

RICHARD S BARR / Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX 75275, Email barr@csvax seas smu edu

(Received August 1990, revised September 1991, November, 1991, accepted April 1992)

Statistical design and analysis of experiments is a cornerstone methodology for scientific exploration and verification that is rarely employed in reporting mathematical software testing This approach is used to study the relative effectiveness of three key algorithms for the reoptimization of network flow problems: primal simplex, dual method, and out-of-kilter Carefully designed experimentation with state-of-the-art codes accompanied by a rigorous statistical analysis isolates the most efficient method(s) for commonly encountered reoptimization problems, and identifies the effect on solution time of problem class, problem size, type of change (bounds, costs, or node requirements), degree and number of changes in parameters, and number of problems in the reoptimization series

When research scientists conduct experiments and subsequently wish to draw inferences from the resulting data, rigorous standards of experimental protocol—including the use of experimental designs—are typically required Unfortunately this is not the norm in the mathematical programming literature, which often uses unreplicated point estimates and no clear design when reporting empirical testing of software The absence of a statistically valid, systematic approach can result in the drawing of unsupportable conclusions regarding the relative performance of alternative algorithms' implementations The lack of an a priori experimental design is one of the main sources of such shortcomings [17 23 26]

This paper illustrates the application of basic principles of statistical design and analysis of experiments to comparing alternative methodologies for the reoptimization of network flow problems While reoptimization is a well-known implementation technique, the wide range of important applications and the lack of a definitive comparison of the leading approaches has prompted this work The sections that follow detail the purposes of the study, the experimental designs, and the analyses of data generated by the solution of nearly 100,000 network problems

1. Network Reoptimization

Reoptimization of a mathematical programming model means to use the solution of one problem to expedite the optimization of another closely related problem. This can be particularly effective when the two solution points are mathematically "close," and when the reoptimizing algorithm can efficiently use a previously obtained solution as a starting point in its search

This classic implementation strategy has application within many optimization methodologies—for large-scale integer, nonlinear, multi-objective, and stochastic programming problems—which require the solution of a series of closely related *network subproblems* Such subproblems differ from each other by one or more problem parameters but have the same network topology, or mathematical structure By using the optimum solution of one subproblem as an advanced starting solution to a subsequent subproblem, substantial reductions in solution time and computational cost can be achieved [21]

The reoptimization process can be applied repeatedly to solve the hundreds or thousands of network subproblems generated within such "higher-level" mathematical programming algorithms This study focuses on its use in solving problems with a pure network structure Described below are several prominent methodologies which require the solution of many related pure network subproblems and hence are amenable to the use of reoptimization (See [6] for a detailed discussion)

- *Branch-and-bound methods* for the fixed charge network problem [11 36 39] This methodology requires the solution of hundreds or thousands of subproblems which all share the same network structure, but differ in the values of arc bounds or costs
- *Dantzig-Wolfe* and *resource-directive decomposition approaches to the multicommodity network problem* [2 24 28 37] In the multicommodity model, k different commodities share arcs in a capacitated network The Dantzig-Wolfe (or price-directive decomposition) method iterates between a master problem and a series of k pure network subproblems, whose costs are modified between iterations The resource-directive approach repeatedly solves a series of pure network subproblems, whose upper bound vectors are modulated Reoptimization can accel-

Subject classifications Statistics, networks Programming

erate the solution of subproblems created by both methods

- *Frank-Wolfe algorithm* and *piecewise linear approximation techniques* for the *convex cost network flow problem* [15 27 32 38] These methods require the solution of many pure network subproblems which have the same network structure and differ only in the objective function coefficients
- *Bender's decomposition method* applied to *multicommodity networks with fixed charges* [13 20] This methodology also iterates between a master problem and a series of network subproblems with an identical network topology Since at each iteration only the demands for individual commodities vary, network reoptimization can be applied for computational advantage
- *Multi-objective programming for the pure network problem* [12 29 35] While there are many approaches to multicriteria optimization, one that can benefit significantly from efficient reoptimization techniques is the *surrogate criterion method*, which associates a weight with each criterion Once a set of weights is selected, a composite "surrogate" objective function is formed to identify a "solution" to the problem Not restricted to a single choice of weights, we can elicit different solutions and a corresponding set of alternative policy scenarios
- *Decomposition methods for stochastic network programming with network recourse* [16 34 41 42] The method iterates between a first-stage problem and a second-stage pure network problem with stochastic node requirements Reoptimization can be applied to this methodology for solving the large number of second-stage pure network problems, which have the same structure but different node requirements
- *Interactive network optimization* [19 25] In many applications of interactive optimization, the user views the solution to a given model, modifies a small portion of the problem, and desires a rapid solution to the modified model Since the previous optimal solution is readily available, it can be used in reoptimization to expedite identification of the new optimal solution

2. Purposes of the Study

To date, a computational comparison of the three best-known pure network algorithms (primal, dual, and out-of-kilter) in the context of reoptimization has not been performed However, researchers have studied—without the aid of statistically designed experiments—one or two of the methods,[1 3 8 12 14 22] and differ in their conclusions regarding network reoptimization Some believe that dual or primal-dual approaches are superior to primal methods for all types of network reoptimization,[3 22] some believe that the dual method is probably better for right-hand-side changes,[8 14] while others emphasize the attractiveness of the primal approach for bounds and right-hand-side changes[1] (see [7] for details)

Our study was designed to resolve these often-contradictory conjectures and conclusions, and answer the following questions

- Is there a best overall method for reoptimizing network problems?
- What are the effects of type and degree of parametric change on the performance of each reoptimization method?
- What are the effects of problem class and size on the performance of each reoptimization method?
- What are the interaction effects on the reoptimization methods when the above factors are changed singly or in combinations?

To fully explore the interaction between algorithms and salient problem characteristics, we (1) developed reoptimization codes for three prominent pure network solution methods, (2) devised a statistical experimental design to evaluate the relative efficiencies of the reoptimization methods, (3) created a portable network reoptimization testing system to generate all necessary data points, and (4) implemented a rigorous statistical analysis of the performance of the algorithms under different experimental combinations

3. Methodologies to be Compared

In this section, we define the type of network problems for which reoptimization is to be studied, outline reoptimization procedures for the primal simplex, dual, and out-of-kilter methods, and describe the computer implementations used in the comparative study

3.1. Problem Statement

Consider a network $G(N, E)$ where N is a finite set of m nodes and E is a finite set of n arcs such that each arc (i, j) is directed from node $i \in N$ to node $j \in N$ Let b_i be the *requirement* at node i, for $i = 1, \ldots, m$, representing supply at a source node if $b_i < 0$ or *demand* at a sink node if $b_i > 0$ An arc directed from its *origin node* $i \in N$ to its *destination node* $j \in N$ is denoted by the ordered pair $(i, j) \in E$ The *flow, cost, lower bound* and *upper bound* of arc (i, j) are represented by x_{ij}, c_{ij}, l_{ij}, and u_{ij}, respectively

Mathematically, the *capacitated minimum cost network flow* or *transshipment problem* (PN) is the following special-structured linear program

$$\text{PN} \quad \text{Minimize} \quad z = \sum_{(i,j)\in E} c_{ij} x_{ij}, \tag{1}$$

$$\text{subject to} \quad \sum_{(i,p)\in E} x_{ip} - \sum_{(p,j)\in E} x_{pj} = b_p,$$

$$p \in N, \quad \text{and} \tag{2}$$

$$l_{ij} \leqslant x_{ij} \leqslant u_{ij}, \qquad \text{for all } (i, j) \in E \tag{3}$$

By substitution of variables, an equivalent to PN with all $l_{ij} = 0$ can be found The remainder of this section's discussion assumes zero lower bounds

Associated with every PN is a corresponding *dual problem*, DN, given as follows

$$\text{DN} \quad \text{Maximize} \quad z = \sum_{i\in N} b_i w_i - \sum_{(i,j)\in E} u_{ij} v_{ij}, \tag{4}$$

$$\text{subject to} \quad w_j - w_i - v_{ij} \leqslant c_{ij}, \quad (i, j) \in E, \tag{5}$$

$$w_i \text{ unrestricted in sign}, \quad i \in N, \text{ and} \tag{6}$$

$$v_{ij} \geqslant 0, \qquad \text{for all } (i, j) \in E, \tag{7}$$

where the m-component vector $\mathbf{w}$ and n-component vector $\mathbf{v}$ denote *dual variables* associated with the conservation-of-flow (2) and the capacity constraints (3), respectively. Associated with each node $i \in N$ is a dual variable w_i, called its node potential. Given arc $(i, j) \in E$, the *reduced cost* is defined as $\bar{c}_{ij} \equiv c_{ij} + w_i - w_j$. We may construct the optimum solution of the *primal problem*, PN, from the optimal solution to the *dual problem*, DN, through the use of the complementary slackness theorem which states that for each arc $(i, j) \in E$

$$\bar{c}_{ij} > 0 \rightarrow x_{ij} = 0, \tag{8}$$

$$\bar{c}_{ij} = 0 \rightarrow 0 \leqslant x_{ij} \leqslant u_{ij}, \tag{9}$$

$$\bar{c}_{ij} < 0 \rightarrow x_{ij} = u_{ij} \tag{10}$$

3.2. Reoptimization Procedures

Given the solution to some PN, changes may be made in the problem parameters so as to create a "closely related" problem which has the same network structure but differs from the original problem in terms of (i) upper bounds ($\mathbf{u}$ is changed to $\mathbf{u}'$), (ii) lower bounds ($\mathbf{l}$ is changed to $\mathbf{l}'$), (iii) node requirements ($\mathbf{b}$ is changed to $\mathbf{b}'$), or (iv) costs ($\mathbf{c}$ is changed to $\mathbf{c}'$). Each of these four cases is treated differently when reoptimizing, as outlined below.

In this section we briefly describe reoptimization procedures for the three algorithmic alternatives primal simplex, dual, and out-of-kilter. (Detailed presentation of these procedures can be found in [1, 3, 4].) Since the primal simplex and dual method specializations for networks are built around the graphical structure of network bases, we first characterize these basic solutions.

3.2.1 Basic Solutions. In simplex-based algorithms for pure networks, the arc-set, E, is partitioned into two subsets basic and nonbasic arcs. To find a *basic feasible solution* for PN, the flows on the nonbasic arcs are set to the upper or lower bounds, and the flows on the basic arcs are uniquely assigned so that constraints (2) and (3) are satisfied. For every basic feasible solution, the node potentials are evaluated such that the complementary slackness equations (8)–(10) are satisfied.

Network theory indicates that each PN basis can be represented as a *spanning tree* of the m nodes with $(m - 1)$ arcs. Given an arbitrary node as the *root node*, the basis is called a *rooted spanning tree*. The root node will be considered to be at the top of the basis tree with all other nodes hanging below it. An optimal solution is obtained when (1) is minimized and constraints (2), (3), and (8)–(10) are simultaneously satisfied.

In describing the reoptimization steps for extreme point methods, it is useful to define the *reduced requirement* at node p, as

$$q_p \equiv b_p + \sum_{(i,p)\in E^l} l_{ip} - \sum_{(p,j)\in E^l} l_{pj} + \sum_{(i,p)\in E^u} u_{ip} - \sum_{(p,j)\in E^u} u_{pj}, \tag{11}$$

where E^l and E^u is the set of nonbasic arcs at their lower and upper bounds, respectively. This is the portion of the requirements represented in the nonbasics.

3.2.2 The Primal Simplex Method Reoptimization Procedures. In reoptimization cases (i) and (ii) involving modified bounds, we assume that the upper bound vector $\mathbf{u}$ and the lower bound vector $\mathbf{l}$ are changed to $\mathbf{u}' \geqslant \mathbf{l}$ and $\mathbf{l}' \leqslant \mathbf{u}$, respectively, where no arc has simultaneous lower and upper bound changes. These changes may destroy the primal feasibility of the optimal basis on hand. Maintaining primal feasibility and obtaining a new optimal solution requires computation of the reduced requirement vector, $\mathbf{q}$, modified for those nonbasic arcs at their changing bound. From the modified $\mathbf{q}$, a new set of basic flows is generated which may not satisfy the bound constraints. Each basic arc with new flow exceeding its upper or lower bound is designated as nonbasic at its violated bound and replaced in the basis with an artificial arc (of appropriate orientation) having a flow of the amount of violation. If artificials have been added, a portion of the node potentials must be updated prior to re-application of the primal simplex method.

In reoptimization case (iii), the node requirements vector $\mathbf{b}$ for a given optimized PN is changed to $\mathbf{b}'$. As before, this change can destroy the primal feasibility of the optimal basis while dual feasibility conditions remain satisfied. To restore primal feasibility efficiently, the reduced requirement vector $\mathbf{q}$ is computed, modified to reflect the changes in node requirements, and used to construct a new set of basic flows. If the resultant basis contains arcs which violate their bounds, appropriate parts of the previous procedure for cases (i) and (ii) are applied to restore primal feasibility and optimality.

In case (iv), where cost changes are introduced, the node potentials are recalculated and the simplex procedure applied to restore optimality. For complete details on all cases, see [1].

3.2.3 The Dual Simplex Method. Reoptimization Procedures. For the dual method, the steps to manage changes in bounds and node requirements are basically the same as those for the primal simplex. The only difference is that basic variable bound violations are handled directly by the dual method.

When changes are made to cost parameters, reoptimization with the dual method is more complicated than with the primal approach. For each *basic* arc with a modified cost (1) substitute an artificial basic arc with a zero upper bound and cost equal to the original cost, and (2) designate the modified arc as nonbasic at its upper (lower) bound if the new reduced cost is negative (non-negative). For each cost-modified *nonbasic* arc at its lower (upper) bound with a new reduced cost less (greater) than zero switch it to its opposite bound and modify the reduced requirements of the arc's incident nodes appropriately With dual feasibility reestablished, a new set of basic flows is then produced from the reduced requirements vector and the dual method is applied to restore primal feasibility and optimality (For a detailed description, see [3])

3.2 4 Out-of-Kilter Method Reoptimization Procedures. Because of its primal-dual approach, the out-of-kilter algorithm accommodates changes in bounds, node requirements, and costs directly Therefore no special procedures are necessary to reoptimize a given pure network problem Since the method only requires a conserving set of flows to begin, problem reoptimization involves simply modifying arc bounds and costs, retaining the previous flows and duals, and applying the algorithm directly For a detailed discussion of the theoretical and implementation of the out-of-kilter method see [9, 18, 27]

3.3. Computer Implementations of the Algorithmic Alternatives

In this section, we present the structural and the functional characteristics of the three primal-, dual-, and out-of-kilter-based reoptimization codes that were tested in this study All codes are written in FORTRAN 77 and require problem data to simultaneously reside in primary (or virtual) storage All programs are based on codes originally written by the same author and, although developed over a wide span of years, contain tight coding and modern data structures

3 3.1. PROPT, a Primal-Simplex-Based Network Reoptimization Code For this study, a primal-simplex-based code, PROPT, was developed for solving and reoptimizing pure network problems PROPT is an extension of the NETSTAR optimizer—a state-of-the-art and improved version of the ARC II code developed by Barr, Glover and Klingman[10]—to which reoptimization routines were added, based on the procedures discussed in Section 3 2 2

PROPT uses the following data structures and node labels to represent the basis tree predecessor, thread, reverse thread, cardinality, last node, node potential, flow, the node requirement, and reduced requirement For a more detailed discussion about the optimization routine and data structures see [10]

3 3.2 DROPT, a Dual-Method-Based Network Reoptimization Code The second reoptimizer code is based on the dual method on a graph [3 14 22 31] Combining the steps of the dual method specialized for networks, the pivoting routines of NETSTAR, and the procedures in Section 3 2 3, an efficient new computer code, DROPT, was developed by the authors for solving and reoptimizing pure network problems Preliminary testing showed the code to be 8–15% faster than the most efficient dual-based code reported to date [3]

Our implementation uses a decreasing-length candidate list of primal infeasible arcs from which the outgoing arc is selected using a smallest-endpoint-cardinality rule (An endpoint node's cardinality is the number of nodes in the basis subtree below and including the node)

In addition to the PROPT data structures, DROPT uses the forward and the backward star structures, to expedite identification of the incoming arc These data permit pricing only a small subset of the nonbasic arcs when performing the ratio test

The reoptimization procedures are built around those discussed in Section 3 2 2 The same programming style is used in both DROPT and PROPT

3 3 3 KROPT, an Out-of-Kilter-Based Network Reoptimization Code The third reoptimizer code, KROPT, is based on SUPERK, an out-of-kilter algorithm code developed by Barr, Glover and Klingman [9] This code forms the optimization portion of KROPT and has been shown to be superior to other out-of-kilter codes by a factor of 2–5 on small- and medium-sized problems and by a factor of 4–15 on large problems Although developed in 1973, it is still considered to be one of the best out-of-kilter implementations to date For a detailed discussion of the out-of-kilter formulation and implementations see [9, 18, 27]

From a programming standpoint, the accommodation of problem changes by KROPT is trivial The preliminary process of obtaining primal or dual feasibility is unnecessary because the solution procedure may begin with a solution which is both primal and dual infeasible Therefore, for reoptimization, the arc parameter modifications are made directly to the problem data and the out-of-kilter algorithm is reapplied, using the previous problem's flows and duals as starting conditions

3 3 4 Construction of a Reoptimization Testing System To simplify and structure the generation and analysis of the experimental data points, a portable network reoptimization testing system (NRTS) was developed NRTS is organized into four components (1) the base problem generator which is the well-known random-network generator, NETGEN[30], (2) the subproblem-series generator, which creates a series of subproblems based on a given base problem and treatment combination, and uses a modified SUPERK, (3) a user-supplied suite of codes to reoptimize the generated subproblem series (here we utilized PROPT, DROPT, and KROPT), and (4) the data analysis module that collects the solution data and performs a statistical analysis to identify the relative efficiencies of the codes (For a comprehensive discussion on NRTS see [4, 5] NRTS is available to the public from the authors)

In summary, for a given run (1) NRTS creates a feasible random base network problem, (2) given a base problem and a set of levels of the experimental factors, cumulative changes are made to the base problem to generate a series of closely related random subproblems, (3) the base problem and subproblems are solved by each of the reoptimizer codes, and (4) solution data are collected in a convenient form for subsequent analysis

4. The Experiment

This section presents the design of an experiment for network reoptimization The experiment, design, implementation and analysis phases are discussed For detailed discussion on the theoretical concepts, principles, and phases of experimental design see [33, 40] Discussions on the experimental design and statistical analysis of computational studies can be found in [4, 17, 23]

In designing our experiment for network reoptimization, the goals were (1) to study the effects of several factors on the time required to reoptimize a series of closely related pure network problems by each of the three reoptimization codes, PROPT, DROPT and KROPT, and (2) to identify the relative efficiency of the three reoptimizers under different combinations of the factors In this manner, the experiment should provide answers to the questions posed in section 2

The *response variable* for comparing the reoptimizers was selected to be the central processing unit (CPU) time required to solve a series of subproblems The total reoptimization CPU time includes the execution of the reoptimization procedures but excludes the input/output processing time and solution time of the base problems

The seven *factors* to be studied in the present experiment are *class of pure network problem, problem size, number of subproblems per series, type of change in problem parameters, percentage change in problem parameters, number of changes per subproblem*, and *type of reoptimizer code* used to solve each subproblem series The experiment studied two classes of pure network problems, transportation and transshipment problems Problem size levels are small (400 nodes and 2,000 arcs), medium (1,000 nodes and 5,000 arcs), and large (1,500 nodes and 8,000 arcs) The factor *type of change* has three levels change in costs, bounds and RHS parameters The *percentage of change in parameter* levels were chosen to be 5% and 25% The *number of changes per subproblem* factor was fixed at 20 In summary, there are seven experimental factors with fixed levels

While the above size dimensions are not large for pure network problems in general, they are of the size encountered in typical reoptimization applications, such as branch-and-bound The *percentage change* levels were chosen to represent applications that tend to yield either few changes from one subproblem to the next, as with depth-first searches, or drastic changes, as with multi-objective programming *Changes per subproblem* was set at a fixed amount to avoid a geometric increase in the number of test cases to be explored Hence our conclusions are generalizable to the extent that these problem types and factor levels match the subproblem characteristics in the application of interest

5. Preliminary Studies

5.1. Number of Subproblems in a Series

Prior to conducting the study's experimentation, it was postulated that the performance of each reoptimizer may be affected by the number of related subproblems that are to be solved as a series The experimentation described below determined the smallest number of series' subproblems for which the hypothesis testing would remain valid, thus minimizing the computational testing effort

5.1.1. The Sampling Method. A random sample of three base problems was selected from among the twelve possible problem combinations of *classes of network problem, problem sizes*, and *percentage changes* The sample problems were randomly selected as a large transportation problem with a 25% change, a medium transshipment problem with 5% change, and a small transportation problem with 5% change For each sample base problem and each type of change, 500 feasible subproblems were randomly generated by NRTS and then solved by each of the three reoptimizers (In two cases involving RHS changes, NRTS was unable to create the full 500 subproblems before problem infeasibility)

The sample consisted of 25 series of 500, one sample of 445 and one sample of 200 reoptimization times Hence, the statistical tests conducted in this section are based on twenty-seven samples with 4,145 generated feasible subproblems and 12,435 subproblems' reoptimization-time observations

5 1 2 Variance Comparisons To verify the effect of the number of subproblems on the overall performances of reoptimizers, two sets of two-way analysis-of-variance (ANOVA) procedures for each type of change over three codes were conducted In the first and second sets of variance analyses, the first 100 and 200 versus the last 400 and 300 CPU times in each sample were considered, respectively In estimating the *population variance* to which the samples belong, the following statistical model is applied

$$X_{ij} = \mu + \alpha_i + \beta_j + E_{ij}, \tag{12}$$

where i is the subproblem number ($i = 1, \quad , 500$), j is the reoptimization code number ($j = 1, 2, 3$), X_{ij} is the CPU time of reoptimizer i for solving the subproblem j, μ is the mean time for all subproblems, α_i is the effect of subproblem i, β_j is the effect of algorithm j, and E_{ij} is the random error effect

The model is utilized as follows The first 100 CPU times and the last 400 CPU times in each sample were considered as two subsamples, making 27 data sets, each consisting of two subsamples For each base problem, type of change, three reoptimizers and two subsamples of 100 and 400 observations, a total of 18 variance analyses were conducted (using SAS GLM) on 300 and 1,200 reoptimization times, respectively, and the estimated variances of subsamples recorded Then, to test for significant differences among the reoptimizers in reoptimizing samples of the first 100 versus the last 400 subproblems, the following hypotheses were established for the variances of the two subsamples

$$H_0 \quad \sigma^2_{100} = \sigma^2_{400} \qquad H_A \quad \sigma^2_{100} \neq \sigma^2_{400} \tag{13}$$

meaning that given a base problem, *type of change* and a reoptimizer, the two normally distributed independent subsamples of 100 and 400 reoptimization times do (H_0) or do not (H_A) have the same variances

To test the hypotheses, the *two-tailed F-test* procedure was employed The F-value is computed by $F = s^2_{100}/s^2_{400}$, where s^2_{100} and s^2_{400} are the mean square errors (estimates of σ^2_{100} and σ^2_{400}) corresponding to subsamples of the first 100 and the last 400 CPU times, respectively The mean

Table I Two-Sided *F*-Test For Comparing Variance CPU Times of Sample Sizes 100 and 400

Type of Change	Sample Problem 1	Sample Problem 2	Sample Problem 3
Cost	MSE_{100} = 0 2452	MSE_{100} = 0 0188	MSE_{100} = 0 0039
	MSE_{400} = 0 1731	MSE_{400} = 0 0249	MSE_{400} = 0 0024
	DF_{100} = 200	DF_{100} = 200	DF_{100} = 200
	DF_{400} = 800	DF_{400} = 800	DF_{400} = 800
	F-Ratio = 1 4167	*F*-Ratio = 0 7550	*F*-Ratio = 1 6250
	SD[a]	SD	SD
Bound	MSE_{100} = 0 2758	MSE_{100} = 0 1552	MSE_{100} = 0 0062
	MSE_{400} = 0 2963	MSE_{400} = 0 0988	MSE_{400} = 0 0048
	DF_{100} = 200	DF_{100} = 200	DF_{100} = 200
	DF_{400} = 800	DF_{400} = 800	DF_{400} = 800
	F-Ratio = 0 9308	*F*-Ratio = 1 5709	*F*-Ratio = 1 2917
	NSD[b]	SD	SD
RHS	MSE_{100} = 0 7302	MSE_{100} = 0 6431	MSE_{100} = 0 0007
	MSE_{400} = 0 4416	MSE_{400} = 0 0074	MSE_{345} = 0 0027
	DF_{100} = 200	DF_{100} = 200	DF_{100} = 200
	DF_{400} = 800	DF_{400} = 800	DF_{345} = 690
	F-Ratio = 1 6535	*F*-Ratio = 87 497	*F*-Ratio = 0 2593
	SD	SD	SD

[a] SD, significantly different variances
[b] NSD, variances not significantly different

square errors were computed from the model described in (12) Table I summarizes the required data to compute *F*-value for testing the hypotheses in (13) A significance level of 5% was selected in advance of the analysis As shown in this table, for all three sample base problems under all three types of changes except one, the two model estimates of corresponding populations' variances are significantly different, implying that the population variances are probably not equal Hence, the null hypothesis in (13) is rejected That is, the effects of reoptimizing a series of the first 100 versus the last 400 subproblems on the codes were significantly different (In the case of bounds changes in sample base problem one, insignificant differences between variances were detected)

The same type of analysis compared the first 200 and last 300 subproblems (see [6] for details) Again, although significant differences in estimates of variances were detected, smaller differences were present in the subsamples of the first 200 versus the last 300 observations In all cases the null hypotheses were rejected, that is, the population variances are probably not equal for two subsamples But since all of the *F*-ratios in this second analysis were close to 1 0 (equal variances) and the sample sizes were large (400 and 600 observations), the equality hypotheses (13) may have been rejected because of the high discriminating power of the *F*-test

From a practical standpoint, knowing that the test of significance involving large samples (like ours) will deem small departures from the null hypothesis as statistically significant, we may make the following conclusion about the variance comparisons in these cases "although statistically significant, the difference between the two estimated variances are too small to be of practical importance, and are ignored in the subsequent analysis "[40] On this rationale, it was decided to select series of 200 subproblems for this experiment

5 1.3 Means Comparison Analysis To verify the previous conclusion, tests of significance among mean reoptimization times were conducted For each sample base problem, type of change and code, multiple comparisons between the mean reoptimization times of subproblems grouped in samples of 100 were performed to identify significant differences between them

To detect significant differences, each sample of 500 CPU times (from a sample base problem, a type of change, and three reoptimizers) is divided into five subsamples of *first 100* through *fifth 100* CPU times Next, Tukey's significant difference tests (within SAS GLM) were conducted to test for equivalence of the five mean times of each algorithm under each type of change

The results of the 27 Tukey tests are summarized in Table II This table shows that in 14 of 27 cases there are no significant differences between mean CPU times of the *first 100* through the *fifth 100* subsamples In 12 of the remaining 13 cases there are no significant differences between the means of *second 100* and *fourth 100* or *fifth 100* subsamples, whereas in six cases, the difference between the *first 100* and *fourth 100* or *fifth 100* subsamples does not appear to be significant

Table II. TSD Test for Five Subsamples' Average CPU Times

Type of Change	Sample Problem	KROPT	PROPT	DROPT
Cost	1	All NSD[a]	1, 2, 3 and 4[b] NSD	All NSD
			1, 3, 4 and 5 NSD	
	2	All NSD	All NSD	All NSD
	3	2, 3 and 4 NSD	2, 3 and 4 NSD	2, 3 and 4 NSD
		3, 4 and 5 NSD	3, 4 and 5 NSD	3 4 and 5 NSD
Bound	1	All NSD	All NSD	All NSD
	2	All NSD	All NSD	All NSD
	3	1, 2 and 3 NSD	2, 3, 4 and 5 NSD	1, 2 and 3 NSD
				2, 3 and 4 NSD
		2, 3, 4 and 5 NSD		3, 4 and 5 NSD
RHS	1	All NSD	All NSD	All NSD
	2	4 and 5 NSD	1, 2, 4 and 5 NSD	2, 4 and 5 NSD
		1, 2, 3 and 4 NSD	3, 4 and 5 NSD	1, 2, 3 and 4 NSD
	3	2 and 3 NSD	1 and 2, 1 and 3,	1 and 2, 1 and 3,
			1 and 4, 2 and 3,	1 and 4, 1 and 5,
		3 and 5 NSD	2 and 4, 3 and 4,	2 and 3, 2 and 4,
			3 and 5 NSD	2 and 5, 3 and 5
		4 and 5 NSD		NSD

[a] NSD, no significant differences among average CPU times of 5 subsamples
[b] 1, 2, 3, 4, and 5 "first 100" through "fifth 100" CPU times subsamples' numbers

There exists only one case for which the differences between the means of *first 100* or *second 100* subsamples are shown to be significant from the *fourth 100* or *fifth 100* subsamples As a result, this means comparison analysis verifies the appropriateness of 200 subproblems per base problem

5.2. Number of Base Problems

Determination of the number of base problems per problem size is another design issue The principles of experimental design call for the minimal number of base problems in order to satisfy two objectives (1) minimize the total computer time for the experiment, and (2) provide an appropriate number of degrees of freedom to study the effects of the whole plot factors—*problem class* and *size*—and their interactions with base problems Having considered the experimental design corresponding to this study to satisfy the two established goals, four base problems per problem size were chosen, thus providing 18 degrees of freedom and allowing the study of the effects of all factors and interactions

6. The Design

The preliminary studies determined that, for each of the 108 experimental conditions, a series of 200 subproblems would be solved for four different base problems This required a total of 432 computer runs, solving 86,400 subproblems

The randomized procedure to generate observations for the various treatment combinations is as follows given a randomly generated base problem, *type of change*, and *percentage change*, a series of 200 subproblems is randomly generated and solved by each of the three reoptimization codes The total reoptimization time for each code is recorded Given a *class of problem*, the order of the experimentation was generate a base problem, randomly determine the *type of change*, randomly select a fixed *percentage change*, randomly generate a series of 200 subproblems and solve by each code

The experiment's characteristics lend themselves to a *split-plot* design (often called *nested* since within each treatment combination there are several treatment subcombinations) The underlying principle of this design is *main plots* —to which levels of one or more factors are applied—are divided into *subplots* or *split plots* to which levels of one or more additional factors are applied Such a scheme reduces the number of observations required and provides more precise information on the subplot factors than on the main plot factors

In this study, the combinations of *class of problem* and *problem size* constitute six main plots, within which the

Table III Characteristics of Transportation and Transshipment Problems

Size	Problem No	No of Nodes	No of Source Nodes	No of Sink Nodes	No of Arcs	Cost Min	Cost Max	Total Supply	No of Transsh SOR	No of Transsh SIN	% Hi Cost	% Arc Cap	Upper Bound Range Min	Upper Bound Range Max	Random Seed No
A Transportation															
Small	P1	400	200	200	2,000	1	1,000	200,000	0	0	0	50	50	100	02135024
	P2	400	200	200	2,000	1	1,000	200,000	0	0	0	50	50	100	46378532
	P3	400	200	200	2,000	1	1,000	200,000	0	0	0	50	50	100	85319210
	P4	400	200	200	2,000	1	1,000	200,000	0	0	0	50	50	100	71685392
Medium	P5	1,000	500	500	5,000	1	1,000	500,000	0	0	0	50	50	100	21328751
	P6	1,000	500	500	5,000	1	1,000	500,000	0	0	0	50	50	100	48597281
	P7	1,000	500	500	5,000	1	1,000	500,000	0	0	0	50	50	100	50832175
	P8	1,000	500	500	5,000	1	1,000	500,000	0	0	0	50	50	100	78530620
Large	P9	1,500	750	750	8,000	1	1,000	750,000	0	0	0	50	50	100	55202473
	P10	1,500	750	750	8,000	1	1,000	750,000	0	0	0	50	50	100	73455831
	P11	1,500	750	750	8,000	1	1,000	750,000	0	0	0	50	50	100	37203644
	P12	1,500	750	750	8,000	1	1,000	750,000	0	0	0	50	50	100	51926435
B Transshipment															
Small	P13	400	25	25	2,000	1	1,000	200,000	0	0	0	50	50	100	61714889
	P14	400	25	25	2,000	1	1,000	200,000	0	0	0	50	50	100	21882420
	P15	400	25	25	2,000	1	1,000	200,000	0	0	0	50	50	100	96293372
	P16	400	25	25	2,000	1	1,000	200,000	0	0	0	50	50	100	49539439
Medium	P17	1,000	50	50	5,000	1	1,000	500,000	0	0	0	50	50	100	51968364
	P18	1,000	50	50	5,000	1	1,000	500,000	0	0	0	50	50	100	61374058
	P19	1,000	50	50	5,000	1	1,000	500,000	0	0	0	50	50	100	52106343
	P20	1,000	50	50	5,000	1	1,000	500,000	0	0	0	50	50	100	10924580
Large	P21	1,500	75	75	8,000	1	1,000	750,000	0	0	0	50	50	100	55204734
	P22	1,500	75	75	8,000	1	1,000	750,000	0	0	0	50	50	100	42048303
	P23	1,500	75	75	8,000	1	1,000	750,000	0	0	0	50	50	100	37203645
	P24	1,500	75	75	8,000	1	1,000	750,000	0	0	0	50	50	100	75215754

combinations of *type of change*, *percentage change*, and *type of reoptimizer* form the subplots

7. The Implementation

Implementation involves the generation of data points to be analyzed within the experimental design All observations were created with NRTS, and standard randomization procedures were used, as follows

First, for each *class* and *size of problem*, four base problems were defined (see Table III, A and B) Next, one of the 108 treatment combinations was randomly selected and, applying NRTS, a base problem and a series of 200 subproblems were randomly generated, solved by the three reoptimizer codes, and the reoptimization times recorded This randomization procedure was repeated until all of the design's data points were produced Replications within *class* and *size* combination were achieved by using different random number seeds

All computational testing was performed on Southern Methodist University's IBM 3081-D24 machine under the VM/CMS operating system The FORTVS2 Fortran compiler and optimization level 3 were utilized Table IV shows the average reoptimization CPU times, in seconds, of four series of 200 subproblems associated with each cell in the split-plot layout (Note that for a few cells, the problem generator was unable to create—after several hundred attempts—a series of 200 feasible subproblems with the combination of factors required)

8. The Analysis

8.1. Statistical Analysis I: Analysis of Variance

Answering the questions posed in Section 2 necessitated comparisons of codes under different treatment combinations defined by the experimental layout This included a comprehensive analysis of the effects of the factors, singly and jointly, on the performance of each code In particular, the objective was to identify the importance of factors and their interactions in terms of the magnitude of their effects on the reoptimization times generated by the codes

The statistical model used to reflect the reoptimization CPU times to the factors and sources of error in this experiment and for the split-plot design is

$$X_{ijklmn} = \mu + K_i + S_j + K_iS_j + P_{k(ij)} + T_l + C_m + R_n + \textit{Subplot factor interactions} + E_{ijklmn}, \quad (19)$$

Table IV Average Reoptimization CPU Times for All Problem Types

Problem Class	Problem Size	Type of Change	Percentage Change	Reoptimization Code Mean CPU Time[a] KROPT	PROPT	DROPT
Transportation	Small	Cost	5	19 38	6 25	12 54
			25	40 17	10 77	24 45
		Bound	5	21 30	10 24	7 23
			25	34 52	16 17	11 23
		RHS	5	27 33	15 59	8 28
			25	n a	n a	n a
	Medium	Cost	5	66.29	19 11	44 66
			25	166 58	39 22	105 94
		Bound	5	69 84	34 03	25 54
			25	133 34	59 91	45 08
		RHS	5	85 72	57 42	28 09
			25	n a	n a	n a
	Large	Cost	5	109 38	32 08	82 71
			25	313 38	72 30	222 95
		Bound	5	119 51	60 21	49 41
			25	213 25	106 64	77 68
		RHS	5	144 65	117 11	53 43
			25	887 78	489 82	264 86
Transshipment	Small	Cost	5	7 98	3 56	6 89
			25	17 40	5 39	17 29
		Bound	5	13 33	9 16	8 39
			25	32 19	17 10	18 97
		RHS	5	n a	n a	n a
			25	n a	n a	n a
	Medium	Cost	5	25 84	9 87	22 58
			25	66 13	15 52	73 60
		Bound	5	42 35	28 32	26 15
			25	89 47	43 19	52 72
		RHS	5	132 15	57 81	64 01
			25	n.a	n a	n a
	Large	Cost	5	38 80	14 74	34 19
			25	116 42	25 92	138 87
		Bound	5	57 94	37 59	36 48
			25	135 83	54 54	82 88
		RHS	5	242 63	104 91	106 95
			25	n.a	n a	n a

[a] Average of four CPU times in each cell, n a = not available (unable to generate 200 feasible subproblems)

where

X_{ijklmn} ≡ the reoptimization CPU time,
μ ≡ the mean CPU time,
K_i ≡ the effect of problem class, $i = 1,2$,
S_j ≡ the effect of problem size, $j = 1,2,3$,
K_iS_j ≡ the interaction of problem class and size,
$P_{k(ij)}$ ≡ the effect of base problem per class i and size j, $k = 1,2,3,4$,
T_l ≡ the effect of type of change, $l = 1,2,3$,
C_m ≡ the effect of percentage change, $m = 1,2$,
R_n ≡ the effect of type of reoptimizer, $n = 1,2,3$, and
E_{ijklmn} ≡ the error term

This model includes 16 subplot-factor interaction terms of two-, three-, four- and five-factor combinations, each of which may affect the response variable

The statistical method required for the analyses was analysis of variance This method provides information for testing simultaneously the significance of the difference between mean reoptimization times of the codes under single or multiple treatment combinations Given the experimental design, the significance of the difference between treatment-combination mean times could be tested by analyzing the variance between the means

The analysis of variance is initiated by a translation of the objectives of the study into statistical hypotheses The hypotheses were categorized into two main groups hypotheses to detect significant difference between single factor means, and hypotheses to identify significant differences between the multiple-factor interaction means All null hypotheses can be stated as the group of population means of reoptimization times under the effects of single or multiple factor treatment combinations are equal The alternative hypotheses are at least two of the means from among the group under the same treatment combinations are not equal The significance level selected prior to the analysis was 5% Table V summarizes the information provided by the ANOVA procedure All null hypotheses were rejected even at a much smaller significance level since the *p*-values are no greater than 0 0001 Thus, at least two of the mean reoptimization times stated in each null hypotheses are significantly different In terms of relative performance, this type of analysis does not permit ranking of the codes under different treatment combinations

8.2. Statistical Analysis II: Comparisons of Means

When comparing more than two means, an ANOVA procedure indicates whether the means are significantly different from each other, but does not show which means actually differ The significances shown by our ANOVA made it desirable to conduct further analyses to determine which pairs or groups of reoptimization average CPU times are significantly different Such comparisons between means are sometimes referred to as *mean comparisons* Rejection of the null hypotheses in Table V necessitates means comparisons to provide detailed information about the observed differences in means

The most commonly used multiple pairwise mean comparison methods are Fisher's Least Significant Difference

Table V Analysis of Variance Table for Reoptimization CPU Times

Source	DF	SS	MS	F	*p*-value
Whole plot					
K	1	192120 87	192120 87	2310 01	0 0001
S	1	2984893 06	492446 53	5921 09	0 0001
K × S	2	199325 49	99662 75	1198 33	0 0001
P(K × S)	18	6667 35	370 41	4 45	0 0001
Split plot					
T	2	360040 72	180020 36	2164 53	0 0001
C	1	540172 94	540172 94	6494 95	0 0001
R	2	292151 15	146075 58	1756 39	0 0001
K × T	2	11194 99	5597 50	67 30	0 0001
K × C	1	159237 71	159237 71	1914 65	0 0001
K × R	2	61098 80	30549 40	367 32	0 0001
S × T	4	181396 83	453492 21	545 27	0 0001
S × C	2	277062 94	138531 47	1665 68	0 0001
S × R	4	144828 32	36207 08	435 35	0 0001
T × C	2	6008844 97	300422 49	3612 23	0 0001
T × R	4	148126 92	37031 73	37031 73	0 0001
C × R	2	108771 64	54385 82	653 93	0 0001
K × S × T	3	8919 02	2973 01	53 75	0 0001
K × S × C	2	3909 66	1954 83	23 50	0 0001
K × S × R	4	35780 42	8945 11	107 55	0 0001
K × T × C	1	3706 40	3706 40	44 56	0 0001
K × T × R	4	16848 06	4212 02	50 64	0 0001
S × T × C	2	5389 22	31946 10	38 41	0 0001
S × T × R	8	88418 70	11052 34	132 89	0 0001
T × C × R	4	263934 37	65983 59	793 38	0 0001
K × S × T × R	6	6079 90	1013 32	12 18	0 0001
K × S × C × R	10	16815 85	1681 59	20 22	0 0001
S × T × C × R	4	3742 92	935 73	11 25	0 0001
K × S × T × C × R	8	5147.22	643 40	7 74	0 0001
Error	252	20958 38	83 19		
Total	359	4748584 79			

procedure, Tukey's Significant Difference (TSD) test and Duncan's Multiple Range test These procedures can compare individual-factor-level means or interaction means Without going through a detailed discussion of alternative methods here, the TSD method was chosen for this study

The TSD procedure was applied to compare and rank the performances of reoptimizers under the effect of different single-factor level as well as treatment combinations Tukey's test controls the *experimentwise error rate* (EER) for multiple comparisons, defined as the probability of rejecting one or more of the null hypotheses when making statistical tests of two or more null hypotheses In this study, having multiple mean comparisons required more control on EER, hence the use of Tukey's test

8.2 1 Example of Four-Factor Means Comparisons The TSD procedure was used to perform a series of analyses, which differed by the number and selection of factors to include Specifically, two-factor through five-factor analyses were conducted, and we illustrate with a four-factor

Table VI TSD Comparisons for Four-Factor-Interaction Mean CPU Times

Problem Class	Problem Size	Type of Change	Sample Size, S	Reoptimization Code Mean CPU Time[a]		
				KROPT	PROPT	DROPT
Transportation	Small	Cost	8	A	B	B
				29 78	8 51	18 49
	Medium		8	A	C	B
				116 44	29 19	75 30
	Large		8	A	B	A
				211 38	52 19	152 83
	Small	Bound	8	A	B	B
				27 91	13 21	9 23
	Medium		8	A	B	B
				101 59	46 97	35 31
	Large		8	A	B	C
				166 38	83 43	63 54
	Small	RHS	8	A	B	B
				27 33	15 59	8 28
	Medium		4	A	B	C
				85 72	57 42	28 09
	Large		4	A	B	C
				516 22	303 47	159 14
Transshipment	Small	Cost	8	A	A	A
				12 51	4 48	12 09
	Medium		8	A	B	A
				45 98	12 69	48 09
	Large		8	A	B	A
				77 61	20 33	86 53
	Small	Bound	8	A	A	A
				22 76	13 13	13 67
	Medium		8	A	B	B
				65 91	35 75	39 44
	Large		8	A	C	B
				96 88	46 06	59 68
	Small	RHS	8	—	—	—
				n a	n a	n a
	Medium		4	A	B	B
				132 15	57 81	64 01
	Large		4	A	B	B
				242 63	104 91	106 95

[a] Averages based on 1,600 or 800 subproblems, those with same letter are not significantly different, n a = not available (unable to generate 200 feasible subproblems)

means comparison (See [4, 7] for details on other comparisons)

To study the effect of experimental combination, the factors *problem class*, *problem size*, *type of change*, and *reoptimizer code* were investigated using the TSD test applied to the joint averages of the factor levels Table VI shows the results of the TSD comparisons using sample size S (S series of 200 subproblems) and experimental error rate of 5% In this table, the performance of each reoptimizer can be ranked relative to the other two codes in the same row by letters A, B, and C Within each row, letters A, B, and C indicate the largest, next largest, and smallest average reoptimization times, respectively, and hence are accordingly associated with the worst, better, and best performance

Also, when two codes have the same associated letter in a given row, their mean times are not significantly different In this case, the performance of the codes are said to be "statistically indistinguishable," and while the mean times are indeed different, they are not significantly so In other words, given the sample size and factors considered, the statistical test cannot determine whether any difference in means is due to the effect of the codes or to sampling error When letters in the same row differ, we indicate this by saying that one code's performance is superior to or dominates another

The four-factor means comparison of Table VI indicates that, *for transportation problems* (1) PROPT was superior to the other two codes on medium and large problems when changes were made to the costs, (2) DROPT was superior for bounds changes, but only on large problems, and for right-hand-side changes on medium and large problems, and (3) for all other combinations, PROPT and DROPT are statistically indistinguishable, and superior to KROPT *On transshipment problems* (1) PROPT again dominated when cost changes were made to medium and large problems, but also on large problems with bound changes, (2) for changes to costs and bounds on small problems, all three codes were statistically indistinguishable, and (3) in all other cases, PROPT and DROPT were top-ranked and indistinguishable

8. Observations and Conclusions

The conclusions to be drawn from the computational data and statistical analysis depend on the number of factors taken into consideration For example, when the two factors *reoptimization code* and *type of change* are involved, the TSD test results summarized in Table VII support the following conclusions

Conclusion A For the reoptimization of the pure network problems tested, in general the dual-based DROPT code dominated the others when changes were made to bounds and right-hand-side values, but the primal-based PROPT code dominated when cost changes were involved

When viewing the three factors *type of change*, *type of problem*, and *reoptimization code*, the results may be summarized in Table VIII and the following conclusions

Table VII Two-Factor Interaction Ranking

Type of Change	Reoptimization Code		
	KROPT	PROPT	DROPT
Cost		●	
Bound			●
RHS			●

● Top-ranked performance

Table VIII Three-Factor Interaction Ranking

Problem Class	Type of Change	Reoptimization Code		
		KROPT	PROPT	DROPT
Transportation	Cost		●	
	Bound			●
	RHS			●
Transshipment	Cost		●	
	Bound		○	○
	RHS		○	○

● Top-ranked performance, ○ Tied for top-ranked performance

Table IX Four-Factor Interaction Ranking

Problem Class	Type of Change	Problem Size	Reoptimizers		
			KROPT	PROPT	DROPT
Transportation	Cost	Small		○	○
		Medium		●	
		Large		●	
	Bound	Small		○	○
		Medium		○	○
		Large			●
	RHS	Small		○	○
		Medium			●
		Large			●
Transshipment	Cost	Small	○	○	○
		Medium		●	
		Large		●	
	Bound	Small	○	○	○
		Medium		○	○
		Large		●	
	RHS	Small	—	—	—
		Medium		○	○
		Large		○	○

● Top-ranked performance, ○ tied for top-ranked performance, — no data available

Conclusion B For transportation problems, the dual-based DROPT code dominated the others when changes were made to bounds and right-hand-side values, but the primal-based PROPT code dominated when cost changes were involved

Table X Five-Factor Interaction Ranking

Problem Class	Problem Size	Type of Change	Percent Change	Reoptimizers		
				KROPT	PROPT	DROPT
A Transportation						
	Small	Cost	5	○	○	○
			25		○	○
		Bound	5	○	○	○
			25		○	○
		RHS	5		○	○
			25	—	—	—
	Medium	Cost	5		●	
			25		●	
		Bound	5		○	○
			25		○	○
		RHS	5			●
			25	—	—	—
	Large	Cost	5		●	
			25		●	
		Bound	5		○	○
			25			●
		RHS	5			●
			25			●
B Transshipment						
	Small	Cost	5	○	○	○
			25	○	○	○
		Bound	5	○	○	○
			25	○	○	○
		RHS	5	—	—	—
			25	—	—	—
	Medium	Cost	5		○	○
			25		●	
		Bound	5		○	○
			25		○	○
		RHS	5		○	○
			25	—	—	—
	Large	Cost	5		●	
			25		●	
		Bound	5		○	○
			25		○	○
		RHS	5		○	○
			25	—	—	—

● Top-ranked performance, ○ tied for top-ranked performance, — no data available

Conclusion C For transshipment problems, DROPT and PROPT are statistically indistinguishable for bound and right-hand-side changes, and PROPT dominates all others when cost changes are made

When adding *problem* size as a fourth factor, the summary Table IX and the statistical analysis yield the following

Conclusion D For small problems, there are insignificant differences between PROPT and DROPT

Conclusion E. For medium and large problems, PROPT dominates when cost changes are made

Conclusion F For medium and large problems, DROPT dominates on bound and right-hand-side changes to transportation problems, but is indistinguishable from PROPT for such changes made to transshipment problems

Different rankings and conclusions emerge when a fifth factor, *percentage change*, is included in the analysis, as summarized in Table X, A and B All of the statistical results yield the following

Conclusion G The out-of-kilter-based code KROPT is never dominant for any combination of factors, although it is occasionally indistinguishable from the other codes when reoptimizing small problems

For researchers, then, the choice of appropriate algorithm depends on the application and its characteristics with respect to the factors studies The results indicate that a combined primal-simplex and dual-method approach to reoptimization would be an optimum methodology to reoptimize closely related pure network problems Since the data structures required to implement each algorithm are similar, and the tree-update operation identical, an integrated approach would be relatively straightforward and would encompass the strengths of both methods

10. Summary

The results given above are generalizable to the extent that the problem characteristics examined match the reoptimization problem of interest, this includes the factors and factor level studied, as well as the use of NETGEN-generated base-problem structures We selected characteristics that hopefully have a broad appeal and widespread application

The reader is encouraged to use this testing process as a model for his or her own experimentation, and to employ rigorous statistical methods in their reporting and decision-making (See Greenberg[23] for further discussion and encouragement) This will not only help elevate the norm in the computational mathematical programming literature to that of the natural, social, and medical sciences, but give the user greater insight into factor effects and interactions in the underlying process, and lend increased confidence in reported results to readers and authors alike

References

1 A ALI, E ALLEN, R.S BARR and J L KENNINGTON, 1986 Reoptimization Procedures for Bounded Variable Primal Simplex Network Algorithms, *European Journal of Operations Research* 23, 256–263

2 A ALI, R. HELGASON, J L KENNINGTON and H LALL, 1980 Computational Comparison Among Three Multicommodity Network Flow Algorithms, *Operations Research 28* 4, 995–1000

3 A ALI, R. PADMAN and H THIAGARAJAN, 1989 Dual Algorithms for Pure Network Problems, *Operations Research 37* 1, 159–171

4 M M AMINI, 1989, *Network Reoptimization A Computational Comparison of Algorithmic Alternatives*, University Microfilms, International, Ann Arbor, Michigan (dissertation, Southern Methodist University, Dallas, TX)

5 M M AMINI and R S BARR, 1989 Network Reoptimization Testing System (NRTS) Users Guide, Technical Report 90-CSE-6, Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX

6 M M AMINI and R S BARR, 1990 Applications of Network Reoptimization, in Proceedings of the 21st Annual Conference of the Southwest Decision Sciences Institute, Decision Sciences Institute, Atlanta, pp 77–78

7 M M AMINI and R S BARR, 1990 Network Reoptimization A Computational Comparison of Algorithmic Alternatives, Technical Report 90-CSE-4, Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX

8 R D ARMSTRONG, D KLINGMAN and D WHITMAN, 1980 Implementation and Analysis of a Variant of the Dual Method for the Capacitated Transshipment Problem, *European Journal of Operational Research* 4, 403–420

9 R S BARR, F GLOVER and D KLINGMAN, 1974 An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes, *Mathematical Programming* 7, 60–87

10 R.S BARR, F GLOVER and D KLINGMAN, 1979 Enhancements to Spanning Tree Labeling Procedures for Network Optimization, *INFOR 17* 1, 16–34

11 R.S BARR, F GLOVER and D KLINGMAN, 1981 A New Optimization Method for Large Scale Fixed Charge Transportation Problems, *Operations Research 29* 3, 448–463

12 G BRADLEY, G G BROWN and G W GRAVES, 1977 Design and Implementation of Large Scale Primal Transshipment Algorithms, *Management Science 24* 1, 1–34

13 G G BROWN, G W GRAVES and M D HONCZARENKO, 1987 Design and Operation of a Multi-commodity Production/Distribution System Using Primal Goal Decomposition, *Management Science 33* 11, 1469–1480

14 A CHARNES and M KIRBY, 1964 The Dual Method and the Method of Balas and Ivancescu for the Transportation Model, *Cahiers du Centre d' Etudies Recherche Operationelle 6* 1, 5–18

15 M COLLINS, L COOPER, R. HELGASON, J KENNINGTON and L LEBLANC, 1978 Solving the Pipe Network Analysis Problem Using Optimization Techniques, *Management Science 24* 7, 747–760

16 L COOPER and L J LEBLANC, 1977 Stochastic Transportation Problems and Other Network Related Convex Problems, *Naval Research Logistics Quarterly 24* 2, 327–336

17 H P CROWDER, R.S DEMBO and J M MULVEY, 1978 Reporting Computational Experiments in Mathematical Programming, *Mathematical Programming 15*, 316–329

18 D R FULKERSON, 1961 An Out-of-Kilter Method for Minimal-Cost Flow Problems, *SIAM Journal of Applied Mathematics 9* 1, 18–33

19 A M GEOFFRION, J DYER and A FEINBERG, 1972 An Interactive Approach for Multicriteria Optimization with an Application to the Operation of an Academic Department, *Management Science 19*, 357–368

20 A M GEOFFRION and G W GRAVES, 1974 Multicommodity Distribution System Design by Benders Decomposition, *Management Science 20 5*, 822–844

21 F GLOVER, D KARNEY and D KLINGMAN, 1974 Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems, *Networks 4*, 191–212

22 F GLOVER, D KLINGMAN and A NAPIER, 1972 An Efficient Dual Approach to Network Problems, *OPSEARCH 9 1*, 1–19

23 H J GREENBERG, 1990 Computational Testing Why, How and How Much, *ORSA Journal on Computing 2 1*, 94–97

24 M HELD, P WOLFE and H CROWDER, 1974 Validation of Subgradient Optimization, *Mathematical Programming 6*, 62–88

25 J HULTZ, D KLINGMAN and G T ROSS, 1979 An Interactive Computer System for Multicriteria Facility Location, Research Report, Analysis, Research, and Computation Inc , Austin, TX

26 R.H F JACKSON, P T BOGGS, S G NASH and S POWELL, 1990 Report of the Ad Hoc Committee to Revise the Guidelines for Reporting Computational Experiments in Mathematical Programming, *Mathematical Programming*, 49

27 J L KENNINGTON and R V HELGASON, 1980 *Algorithms for Network Programming*, John Wiley & Sons, New York

28 J L KENNINGTON and M SHALABY, 1977 An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems, *Management Science 23 9*, 994–1004

29 D KLINGMAN and J MOTE, 1979 Solution Approaches for Network Flow Problems with Multiple Criteria, Research Report CCS 363, Center for Cybernetic Studies, University of Texas, Austin, TX.

30 D KLINGMAN, A NAPIER and J STUTZ, 1974 NETGEN A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems, *Management Science 20 5*, 814–821

31 C C LEMKE, 1954 The Dual Method of Solving the Linear Programming Problem, *Naval Research Logistics Quarterly 1*, 36–47

32 H S MAHMASSANI and K C MOUSKOS, 1989 Vectorization of Transportation Network Equilibrium Assignment Codes, in R Sharda, B Golden, E Wasil, O Balci, W Stewart (eds), *Impacts of Recent Computer Advances on Operations Research*, Elsevier Science Publishing Co , New York, pp 71–81

33 R.L MASON, R F GUNST and J L HESS, 1989 *Statistical Design and Analysis of Experiments*, John Wiley & Sons, New York

34 J M MULVEY and H VLADIMIROU, 1991 Solving Multistage Stochastic Networks An Application of Scenario Aggregation, *Networks 21*, 619–643

35 D OLSON, B SHETTY and M VENKATARAMANAN, 1989 Network Reoptimization Procedure for Multiobjective Network Problems, *Annals of Operations Research 20 1*, 219

36 U D PALEKAR, M H KARWAN and S ZIONTS, 1990 A Branch-and-Bound Method for the Fixed Charge Transportation Problem, *Management Science 36 9*, 1092–1105

37 D PHILLIPS and A GARCIA-DIAZ, 1990 *Fundamentals of Network Analysis*, Waveland Press, Prospect Heights, IL

38 Y SHEFFY, 1985 *Urban Transportation Networks*, Prentice-Hall, Englewood Cliffs, NJ

39 B SHETTY, 1990 A Relaxation/Decomposition Algorithm for the Fixed Charge Network Problem, *Naval Research Logistics Quarterly 37 2*, 327

40 G W SNEDECOR and W G COCHRAN, 1976 *Statistical Methods*, The Iowa State University Press, Ames, IA

41 S W WALLACE, 1986 Solving Stochastic Programs with Network Recourse, *Networks 16*, 295–317

42 S W WALLACE and R J-B WETS, 1989 Preprocessing in Stochastic Programming The Case of Uncapacitated Networks, *ORSA Journal on Computing 1 4*, 252–270