CERIAS Tech Report 2001-129
The Problem of Highly Constrained Tasks in Group Decision Support Systems
by J Rees, R Barkhi
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

ELSEVIER

European Journal of Operational Research 135 (2001) 220–229

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Theory and Methodology

The problem of highly constrained tasks in group decision support systems

Jackie Rees [a,*], Reza Barkhi [b,1]

[a] *Krannert Graduate School of Management, Purdue University, West Lafayette, IN 47907-1310, USA*

[b] *Department of Accounting and Information Systems, Pamplin College of Business, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA*

Received 25 January 2000; accepted 13 November 2000

Abstract

Most experimental uses of group decision support systems (GDSS) are associated with relatively unrestricted domains, for example, idea generation and preference specification, where few restrictions on potential solutions exist. However, an important GDSS task is that of resource allocation across functional areas of the organization, including supply chain applications. These types of tasks, such as budget planning and production planning, are typically highly constrained and difficult to solve optimally, necessitating the use of decision aids, such as those found in GDSS.

We use a model based on adaptive search of a genetic algorithm as the analogy for the group decision making process. We apply this model to experimental data gathered from GDSS groups solving a production planning task. The results indicate very low estimated crossover rates in the experimental data. We also run computational experiments based on adaptive search to mimic the GDSS data and find that the low estimated crossover rate might be due to the highly constrained search space explored by the decision making groups. The results suggest further investigation into the presumed beneficial effects of group interaction in such highly constrained task domains, as it appears very little true information exchange occurs between group members in such an environment. Furthermore, the simulation technique can be used to help predict certain GDSS behaviors, thus improving the entire GDSS process.

Keywords: Group decisions and negotiations; Genetic algorithms; Evolutionary computations

1. Introduction

Group decision support systems (GDSS) are intended to support meetings and facilitate group work in the organization. Specifically, Gallupe and DeSanctis (1988) describe the purpose of GDSS to be the promotion of interaction and information exchange among parties with conflicting goals and

* Corresponding author. Tel.: +1-765-494-0320; fax: +1-765-495-9658.

E-mail addresses: jrees@mgmt.purdue.edu (J. Rees), reza@vt.edu (R. Barkhi).

[1] Tel.: 540-231-5869; fax: 540-231-2511.

0377-2217/01/$ - see front matter
PII: S0377-2217(00)00323-4

priorities. This support for group interaction is provided through communication support, process structuring and decision tools. Sample applications of GDSS include quality team support at the IRS (DeSanctis et al., 1992), and at IBM (Grohowski et al., 1995). These applications are often strategic in nature, thus categorizing the task at hand as preference tasks (McGrath, 1984), where the number of possible alternative outcomes is relatively large. Other popular uses of GDSS are for idea generation or brainstorming tasks. These applications are also relatively unconstrained, often limited to just the imagination of the participants.

Other GDSS tasks have serious implications for not only the organization but for the entire supply chain. These tasks include resource allocation tasks such as budget planning and production planning. These tasks can be classified as mixed-motive negotiation tasks (McGrath, 1984). Typically, participants have mixed-motives to compete and cooperate, and hence, the interests of all parties cannot be accommodated. One feature distinguishing these types of tasks from others is the highly constrained nature of the search space. Examples of these tasks include budgeting and order fulfillment both subject to resource constraints such as capacity limitations or financial limitations. Within the context of Simon (1977) view that managerial decision making is the search for optimal or near-optimal solutions through a solution space, many combinations (budget items, production schedules) are infeasible thus requiring increased considerable overhead in the search process.

As stated above, the purpose of GDSS is to facilitate interaction and information exchange between group members. However, in such highly constrained GDSS tasks, such as budgeting or production planning applications, the question arises of how much interaction and information exchange actually occurs between GDSS users and is that actual level of interaction and information exchange appropriate to the desired GDSS outcome?

Rees and Koehler (1999) have developed an evolutionary model based on genetic search that we use to describe adaptation in groups using GDSS. We apply this model to data generated from experimental GDSS sessions. The experiments examined the application of GDSS to a highly constrained task, namely a production planning problem (Barkhi, 1995). This problem is of considerable importance given the high dollar impact such problems have on the organization and on the entire supply chain. We use the experimental data to estimate the parameters for a genetic search model for this problem. Subsequently, we use these genetic parameters to gain insight into the levels of interaction and information exchange between GDSS group members for the particular GDSS experiments. Our objective is to show that these parameters can be used to develop adaptive search agents that mimic the pattern of decisions made by experimental GDSS groups. In addition, we examine the impact of the highly constrained search space on the genetic operator guiding information exchange and provide possible courses of action to improve the search process.

The remainder of the paper is organized as follows: Section 2 examines the background literature on the use of GDSS on such highly constrained problems. The role of evolutionary modeling in the study of similar systems is discussed in this section. An overview of the genetic algorithm as it is used in this context is also provided. Section 3 discusses the evolutionary model and the roles of the genetic operators, selection, crossover and mutation. Section 4 compares the estimated crossover rates from GDSS data sets to the simulated outcomes of such experiments. Finally, Section 5 presents the conclusions resulting from the comparisons and states recommendations for GDSS use in such problems as well as discusses future research directions.

2. Literature review

2.1. GDSS tasks

The task addressed by GDSS users is obviously an important variable in the application of such systems (Lam, 1997). The majority of GDSS experiments have utilized or focused on preference

and idea generation tasks. As described above, these tasks are relatively unconstrained; few restrictions are placed on the search process. Less attention has been focused on highly constrained tasks. However, there exist several examples of the application of GDSS to resource allocation type tasks. Barkhi (1995) experimentally examined the use of GDSS in a production planning setting. Nunamaker et al. (1991) examined product design and Dennis et al. (1995) used budget creation as the GDSS task. These tasks are considered representative of actual GDSS use in organizations (Gillenwater et al., 1995) and are typically highly constrained. Many of the possible solutions (or "points") in the search space are infeasible given the constraints on the task. Additionally, these tasks are complex enough to mandate the use of decision tools to model and solve such tasks. Within this context, GDSS is intended to not only provide decision support but to also facilitate the "...dynamic exchange of decisional information" for such tasks (Gillenwater et al., 1995).

2.2. Evolutionary models

The need to better understand the underlying processes within groups and other forms of organizations is not restricted to GDSS use. Computational models based on evolutionary algorithms, such as the genetic algorithm have been applied in several areas. Bruderer and Singh (1996) used a genetic algorithm to model organizational evolution. This evolutionary model set the stage for creation of a simulation model allowing greater insights into the rise and fall of organizations given specific characteristics. A genetic algorithm was the model for group interaction and information exchange within groups of artificial agents solving rule induction problems (Sikora and Shaw, 1996). Others have applied GAs to groups of agents solving organizational problems (Marks et al., 1995; de la Maza and Yuret, 1995). These examples demonstrate the potential utility in using an evolutionary approach in modeling social processes. The next section describes the genetic algorithm as it will apply to the remainder of the paper.

2.3. Genetic algorithms

Genetic algorithms are general-purpose search algorithms driven by the basic principles of Darwinian natural selection and evolution. Search is performed from a population of agents, rather than the traditional single point. Such agents, called strings or chromosomes, explore a space using three basic operations. First, strings are evaluated according to a given objective function. This evaluation, or fitness, influences the proportion of the string in the next time series, or generation. Fitter strings generally have a greater chance of being stochastically selected for the next generation. Second, selected strings are recombined, or crossed, in hopes of discovering better or fitter strings by combining genetic material. Third, the selected strings are randomly mutated to replace any lost diversity after selection and crossover. As such, GAs are a stochastic search technique.

The simple genetic algorithm uses three biologically inspired operators. Selection, or reproduction, is the operator that tends to take the "fittest" members of the population for use in generating the next generation. Crossover combines genetic material between selected members of the population. Finally, mutation adds diversity back into a population.

Selection occurs similar to that of asexual reproduction in the natural world. Chromosomes that are deemed "fit" by measure of a pre-defined fitness function are stochastically more likely to be represented in future populations. Strings are thus drawn with replacement from the current generation with bias according to a fitness measure and placed into the next generation. This method is known as stochastic sampling with replacement, or more commonly referred to as "roulette wheel" selection. Other commonly used selection schemes include tournament selection and rank selection. In tournament selection, strings are drawn from the population using the method above in pairs and the string with the higher fitness value is placed in the new population. Rank selection starts by sorting the population according to fitness value. Each string receives new copies that are placed in the new population according to a

function of this ordering. Several other variations of selection are discussed in Goldberg (1989).

Crossover implements a mating strategy for the combination of "good" genetic material between fit parents. After the selection procedure is complete, crossover is applied with a predetermined, fixed probability, called the crossover rate, usually ranging from 0.6 to 1.0. Two members of the new population are paired up, each selected with the probability given by the crossover rate. In the most common crossover scheme, one-point (also called single-point) crossover, a single site is uniformly selected with probability $1/(\ell - 1)$, where ℓ is the length of the string. The parent sub strings are exchanged on the right-hand side of the crossover site. Two-point (also called multi-point) crossover works in a manner similar to that of one-point crossover. However, the string is viewed as a ring and two crossover sites are randomly and uniformly selected. The sub strings demarcated by the two points are thus exchanged. Uniform crossover works slightly differently. Each string is selected for crossover just as in single-point crossover, but rather than selecting a crossover site, each bit in the string is exchanged with the corresponding bit in the other string with probability $2^{-\ell}$ (Vose, 1999). Other crossover schemes exist and are discussed in Goldberg (1989).

Mutation is the last operation on the population before the next generation is completely formed. In the binary case, mutation simply requires the mutated bit becomes its complement, i.e., 0 becomes 1 and vice versa. Under uniform mutation, mutation is applied with a fixed, predetermined probability to each gene (each bit) in every string. The mutation rate is kept very low, usually between 0.001 and 0.005, in order to keep the search from diversifying too rapidly. Other mutation schemes are available, see Goldberg (1989).

3. The evolutionary model for GDSS

The evolutionary model for GDSS is discussed in its entirety in Rees and Koehler (1999). The foundation of the model is the genetic algorithm. This evolutionary model captures the adaptive processes undergone by the group as it uses the GDSS. The underlying principle of the model is that the problem-solving process of GDSS groups can be mimicked by a genetic algorithm utilizing *selection*, *crossover*, and *mutation* (Rees and Koehler, 1999). The model is especially suited to highly structured task domains, such as production planning problems.

The implementation of the model can be described as follows: the ideas or proposed solutions exchanged between group members using the GDSS can be encoded as strings. These strings can then be temporally grouped into populations or generations of strings. The sizes of the populations are variable and are a function of the interaction between group members. The fitness function for the group can take many forms. An incentive scheme can easily be encoded into a fitness function, as can a voting scheme or perhaps a utility function. The model assumes the genetic algorithm has three operators: *selection*, *crossover*, and *mutation*. The role and implementation of each of these operators are discussed below.

The purpose of the *selection* operator is to identify better or "fitter" solutions in accordance with the fitness function and insert these strings into the next generation. Ensuring the "survival of the fittest" is the role of the selection operator. The specific implementation of selection may vary from application to application (and from task to task). However, rank selection appears to be a generally robust selection operation. Rank selection, considered a non-parametric procedure, sorts the strings in the population according to fitness value. Copies of individual strings are inserted into the next generation according to a function of the original ranking. Essentially, the higher-ranked the idea, the more likely it will influence subsequent generations.

The role of the *crossover* operator is to combine "genetic" material or information between two solution strings. For this reason the crossover operator is often called the "exploitation" operator. Two strings are mated with probability χ (the crossover rate). Uniform crossover appears to be a fairly good implementation choice for the evolutionary model for GDSS (Rees and Koehler, 1999). Uniform crossover works by moving

bit-wise down the pair of strings, exchanging bits with probability χ. The appeal of uniform crossover is the ability to exchange a variable number of information segments between the string pairs, which is a more dynamic approach than either single-point or multi-point crossover (Goldberg, 1989).

The role of the *mutation* operator is to prevent the search from becoming trapped at local optima. For this reason the mutation operator is often called the "exploration" operator (Goldberg, 1989). Bits of information along the strings are randomly altered at a pre-defined mutation rate, μ. The mutation operation adds diversity to the search by adding random information back to the solution strings. Uniform mutation is the implementation of the mutation operator (Rees and Koehler, 1999). Under uniform mutation, mutation is applied with a fixed, pre-determined probability to each gene (each bit) in every solution string. The mutation rate is kept very low, usually between 0.001 and 0.5 to prevent the search from diversifying too rapidly.

One of the advantages to using a GA as the basis for an evolutionary model is that a large body of mathematical knowledge exists about the GA (Rees and Koehler, 1999). We use experimental data from GDSS sessions (Barkhi, 1995), and apply maximum-likelihood estimation techniques to learn GA parameter settings for mutation and crossover rates. This allows us to mimic the decision process of GDSS users and estimate the trajectory of their decision outcomes. In other words, we use adaptive search and tailor it to specific GDSS groups incorporating the decision process of the group into the design of the adaptive search. In the next section we summarize the mathematical models for the simple GA and show how these can be used to determine a maximum-likelihood estimate for a GDSS trajectory.

The model can be exactly described as follows: Let Ω be a collection of binary strings of length ℓ and let $r = |\Omega| = 2^{\ell}$ be the number of possible strings. These strings can be equivalently considered as the integer equivalents $0, 1, \ldots, r-1$. Let P be a population of elements from Ω, where $n = |P|$ is the population size and N is the number of possible populations. N is computed by the formula

$$N = \binom{n+r-1}{r-1}. \tag{1}$$

A population is a multi-set meaning: it may contain multiple copies of the same string. Consider the Markov chain where the possible populations of size n are the states. Express a state by the vector of length r, ϕ_i, having as its kth component the number of copies of string k in the population. Let e be a vector of 1's and e' its transpose. Each ϕ_i is defined by

$$e'\phi_i = n, \tag{2}$$

$$(\phi_i)_j \in \{0, 1, \ldots, n\}, \quad j = 0, 1, \ldots, r-1. \tag{3}$$

The transition probabilities from state (population) i to j are computed by

$$P_{i,j} = n! \prod_{g=0}^{r-1} \frac{q_{i,g}^{(\phi_j)_g}}{(\phi_j)_g!}, \tag{4}$$

where

$$q_{i,g} = \mathscr{M}(\mathscr{F}(\phi_i))_g. \tag{5}$$

Vose (1999) uses $\mathscr{F}$ to capture the selection process and $\mathscr{M}$ the mixing operators (mutation and crossover). In particular

$$\mathscr{M}(x)_i = (\sigma_i x) M \sigma_i x, \tag{6}$$

where the permutation of x, $\sigma_k x$, is defined by

$$\sigma_k x = \begin{pmatrix} x_{k\oplus 0} \\ \vdots \\ x_{k\oplus(r-1)} \end{pmatrix}. \tag{7}$$

Let $M_{g,k}$ be the probability that the string of all zeros is the child of the mating process between parent strings g and k (where g and k are the integer values corresponding to the strings). A general form of the mixing matrix, M, was given by Vose and Wright (1995) as

$$M_{x,y} = \sum_{j,k} \mu_j \frac{\chi_k + \chi_{\bar{k}}}{2} \delta(x \otimes k \oplus \bar{k} \otimes y = j). \tag{8}$$

Here μ_j and χ_k are called mutation and crossover masks and $\delta(x)$ is 1 when x is true and 0 otherwise.

The various mutation and crossover schemes can be captured using appropriate choices for these masks. For example, letting

$$\chi_i = \begin{cases} \chi c_i & \text{if } i > 0, \\ 1 - \chi + \chi c_0 & \text{if } i = 0, \end{cases} \tag{9}$$

with $c_i = 2^{-\ell}$ giving uniform crossover.

For uniform mutation we have

$$\mu_i = (\mu)^{e'i}(1-\mu)^{\ell - e'i}, \tag{10}$$

where $e'i$ is the number of non-zero bits of i.

The selection process is captured through $\mathscr{F}$. Rank selection is given by

$$\mathscr{F}(\phi)_i = \int_{\sum_{(\phi)_j \delta(f(j) < f(i))}}^{\sum_{(\phi)_j \delta(f(j) \leqslant f(i))}} \rho(y)\, \mathrm{d}y, \tag{11}$$

where ρ is any continuous increasing probability density over $[0,1]$ (see Vose, 1999).

These models can be easily extended to varying-sized populations as follows. Let $P_{i,j}(I,J)$ be the probability of going from state i (where populations are of size I) at time t (the current generation) to state j (where populations are of size J) at time $t+1$. Then we have

$$P_{i,j}(I,J) = I! \prod_{g=0}^{r-1} \frac{q_{i,g}^{(\phi_j)_g}}{(\phi_j)_g!}, \tag{12}$$

where

$$e'\phi_i = I, \tag{13}$$

and

$$e'\phi_j = J. \tag{14}$$

Given an observed trajectory of a GA process, we wish to estimate the underlying parameters used by the GA. That is, we wish to estimate rates χ and μ, the likelihood of an observed trajectory is proportional to the product of the transition probabilities along the path. Hence, the likelihood of a given chain going from j_1 to j_2 to $j_3 \ldots$ is

$$P_{j_1,j_2}(J_1,J_2)P_{j_2,j_3}(J_2,J_3)\cdots P_{j_{T-1},j_T}(J_{T-1},J_T), \tag{15}$$

where $J_1, J_2, \ldots, J_T$ are the population sizes at times $t = 1, 2, \ldots, T$. We use a simple maximum-likelihood procedure in deriving estimates. Maximum-likelihood estimators have several desirable properties, including invariance, sufficiency (if the parameter itself is sufficient) and efficiency (Mood, 1950). To find the maximum-likelihood estimate for each parameter of interest, namely crossover, χ, and mutation, μ, we maximize the likelihood function given above. Therefore, we must solve

$$\max_{\chi,\mu} \prod_{i=1}^{T-1} P_{j_i,j_{i+1}}(J_i, J_{i+1}), \tag{16}$$

where T is the number of observed populations and J_i is population i's size.

In order to find the maximum-likelihood estimate for our Markov chain, we could set the partial derivatives with respect to the mutation and crossover operators to zero and solve for μ and χ. The partial derivatives, with respect to χ, are relatively easy to derive but those for μ are highly non-linear. Furthermore, it is unlikely that first-order conditions would be sufficient. Besides, it appears the equations would be nearly impossible to solve. Therefore, an approximately exhaustive search over a grid should be performed to determine the (near) optimal values of the crossover and mutation rates. An iteration through the values of μ from 0.0 to 0.5 (where 0.5 represents a random search in the binary case) and the values of χ from 0.0 to 1.0, inclusive, is appropriate, in accordance with typical GA practice.

4. The search problem

As discussed in Section 2, much of the prior experimental research on GDSS has focused on tasks that are not highly constrained. The implications of such highly constrained tasks on information exchange amongst GDSS group members have not been well studied. For example, do GDSS groups faced with highly constrained tasks exchange information the same way as GDSS groups faced with tasks that have few constraints? If we assume that GDSS groups that face less-constrained tasks are able to ex-

change parts of ideas, or idea segments to build better or more satisfactory ideas or solutions, do GDSS groups facing more constrained tasks also exchange these information segments? Or are they required to exchange entire ideas, as combining parts of ideas would typically result in infeasible idea strings? Using the model described in the previous section, it is possible to gain insights into this GDSS process and others using such tasks. Among the studies that have been performed on highly constrained tasks, Barkhi (1995) study examined groups faced with a resource allocation problem, in this case a production planning problem. The task faced by the experimental groups was to determine the optimal set of customer orders to fill, given revenue and cost information, subject to departmental capacity constraints, and departmental cost variation.

4.1. Task details

Four products are each made to order. Since each product is customized to customer specifications, each customer order provides unique revenue. There are three functional areas, marketing, production and purchasing, each having one manager. Each manager's task is to decide which customer orders to fill and how much departmental effort to expend in filling each order. If an order is selected for filling, the order must be completely filled, in other words no partial orders are allowed. A table containing 20 customer orders, including products and their quantities was provided to group members. Also, the expected revenues from each customer order were provided to the group members.

Each manager was provided an internal (as in internal to the specific department) description of the uncompensated departmental costs (UDEC) and actual departmental costs (ADC) associated with each customer order. Each manager needed to choose which level of effort to expend for filling a specific customer order. The effort level was discretized into four levels and the UDECs and the ADCs were provided at each level. As can be expected, as the UDEC increases, the ADC decreases but not necessarily in a linear fashion. Also, for each order, the projected costs (PC), the cost estimate each department has for filling the order, were provided. Each department manager also received a set of capacity limits for each product. Each department receives the projected costs for every other department within the firm, but UDEC and ADC are considered internal to each department and are only provided to outside departments (including the leader) at the discretion of the individual departmental manager. The departmental managers could elect to provide the leader with false information regarding these costs, withhold the figures or provide accurate estimates of these departmental costs. This simulated manufacturing company produces K different products. Each customer order specifies the quantity of each of the products requested. There are S customer orders, each with different profit implications and product quantity requirements. The problem presented to each group is to decide which subset of these S orders to fill with the objective of profit maximization considering that manufacturing capacity cannot be violated.

The terminology:

D	number of departments
K	number of products
S	number of customer orders
E	number of effort levels
PC_{id}	projected cost of filling order i at dept. d
ADC_{ijd}	actual departmental cost of filling order i expending effort j at department d
UDEC_{ijd}	uncompensated departmental effort cost is cost that the department d incurs for filling order i at effort level j
Rev_i	revenue generated by filling order i
X_{ij}	$\begin{cases} 1 & \text{if order } i \text{ taken at effort level } j, \\ 0 & \text{otherwise} \end{cases}$
Y_i	$\begin{cases} 1 & \text{if order } i \text{ taken,} \\ 0 & \text{otherwise} \end{cases}$
Q_{ik}	quantity of product k requested by order i
C_k	production capacity of product k

The problem that each group member had to solve can be modeled as follows:

$$\max \quad \sum_{i=1}^{S}\sum_{j=1}^{E}\left[\left(\mathrm{Rev}_i - \sum_{d=1}^{D}\mathrm{ADC}_{ijd}\right)(0.15) - \mathrm{UDEC}_{ijd}\right]X_{ij} \tag{17}$$

$$\text{s.t.} \quad \sum_{i=1}^{S} Q_{ik}Y_i \leqslant C_k \quad \text{for } k = 1,\ldots,K.$$

The data from the experiments (Barkhi, 1995), including the solutions proposed (and the order in which they were proposed) and the group decisions, were used to validate the model. The experiments were performed on 50 experimental groups (although data from two groups were later discarded), where each group was comprised of upper-classmen business school students. The subjects were rewarded with class credit in accordance to performance in the experiments. The GA model was applied using rank selection, uniform crossover and uniform mutation. This yielded the estimated or "best fit" crossover and mutation rates from the experimental data. As the groups had already "solved" the production-planning problem, feasibility in terms of the crossover operator was not an issue, although it was in the simulation described below. The estimated crossover and mutation rates from the experimental data are provided in Table 1. The estimated rates were computed for each experimental group (48 total) and averaged to get overall crossover and mutation rates. The population sizes were dynamic, meaning we allowed the population size to vary as a function of group member interaction (Rees and Koehler, 1999).

One thing to notice is that the estimated crossover rate seems rather low compared to traditional GA settings, usually ranging from 0.6 to 1.0. The estimated mutation rate does not raise any such flags. To shed more light on this issue, we conducted computational experiments by simulating the GDSS task with the adaptive search technique. The parameters of the adaptive search were derived from the experimental data (Barkhi, 1995).

Table 1
Estimated crossover and mutation rates using rank selection

Average estimated uniform crossover rate	Average estimated uniform mutation rate
0.121	0.029

4.2. Simulation details

A simulation was developed to better understand the resulting low estimates of the crossover parameter. Group member "agents" were created, where each agent had access to the order information as the respective human group members. Each agent can generate a solution to the problem, which can be encoded as a string of 20 binary digits. A "1" indicates that a particular customer's order is to be included in the set of orders to be filled, whereas a "0" indicates that a particular customer's order is to be excluded from the set of final orders. The agents are uniformly chosen with replacement to contribute their solution to the rest of the group members. After four solutions have been generated, the solutions are selected, crossed and mutated in accordance with the GA settings. The process then repeats itself for five generations. Five was chosen due to its consistency with the number of solutions generated by the experimental GDSS groups in Barkhi (1995) study.

4.3. Results

The groups' average fitness was maximized between $0 < \chi < 0.4$ and $0.001 < \mu < 0.01$. The summarized findings are presented in Table 2. Each row in the table indicates the average fitness (through simulation) for each possible setting of crossover and mutation rates. These findings are fairly consistent with the estimated crossover and mutation rates found from applying the evolutionary model to the experimental data. Specifically, this finding indicates that the genetic algorithm was not able to take full advantage of the crossover operator, most likely due to the highly constrained nature of the search problem, that is very few combinations of orders were feasible. This resulted in many of the crossed-over string pairs being unusable in solving the problem.

Table 2
Computational results for selected crossover and mutation rates

Uniform crossover rate	Uniform mutation rate	Average fitness value
0.1	0.001	75.375
0.1	0.01	63.75
0.1	0.1	65.375
0.3	0.001	75.375
0.3	0.01	63.75
0.3	0.1	68.438
0.7	0.001	53.625
0.7	0.01	63.75
0.7	0.1	63.938
1.0	0.001	61.25
1.0	0.01	63.75
1.0	0.1	70.375

This also indicates that exploration via the mutation operator becomes more critical to the success of the search as opposed to the traditional crossover operator. An interesting possibility suggested by the simulated data set is the potential of an interaction effect occurring between the various combinations of crossover and mutation rates.

As the behavior of the interaction between the crossover and mutation operators is not completely understood within the GA community, we will leave this possibility as potential for future research.

5. Conclusions

The major finding of this research was that the information exchange between GDSS group members for this particular task was mostly limited to the exchange of entire solution strings, rather than exchanging parts of solution strings, which inherently limits the information exchange capabilities of GDSS use. It appears that the GDSS users for this task relied essentially on enumerative search, where the generation of points was provided by the particular decision aid employed, rather than engaging in an intelligent search. This finding leads us to re-affirm that the appropriate GDSS tool is needed for the problem and the context and that there should be task-technology fit (Zigurs and Buckland, 1998). This in itself was nothing new. The new and relevant finding was the problem space can be so constrained as to mitigate the so-called "helpful" effects of GDSS the most important of which is fostering information exchange. Therefore, this finding needs to be taken into account when selecting GDSS tools. The implication is that the specific decision support tool is very important not only in generating feasible solutions but also in guiding the users towards the best possible solution for the group as a whole and each of the GDSS users.

Additionally, this finding calls into question the lack of appropriate GDSS simulation tools. The simulation performed in this experiment is quite simplistic. However, the findings, while preliminary, are insightful and provide evidence that GDSS users are not exchanging as much information as previously assumed. Much could be done to create a fairly sophisticated simulation tool that would be helpful to GDSS researchers in studying different GDSS configurations, such as group size, task, incentive structure and other variables.

Future research calls for enhancing the current GA model to examine various implementations of crossover and mutation operators. One possibility discussed in Section 2 includes using crossover and mutation masks, allowing the best crossover and mutation operator implementation to be computationally determined for a particular data set. Additionally, more GDSS experiments examining these types of constrained problems need to be undertaken to gain further insight into the process. These tasks can certainly be extended across the supply chain and to long-term planning type tasks. Also, more comparative research needs to be performed on relatively unconstrained tasks. This research would provide another useful benchmark for this and future, related study.

Another avenue of research is that of adaptive GDSS, incorporating adaptive GA-fueled agents into GDSS. By incorporating adaptive agents into the system, the agents could act as additional problem solvers. The solutions generated by the agents could be provided to the appropriate group members, thus expanding the search for each user. Whether or not this improvement in the search process would lead to improved outcomes is uncertain at this point. However, the demand for

improved problem-solving techniques, especially for technology-assisted groups is only increasing, thus more research needs to be performed on such issues.

References

Barkhi, R., 1995. An empirical study of the impact of proximity, leader, and incentives on negotiation process and outcomes in a group decision support setting. Unpublished doctoral dissertation. The Ohio State University.

Bruderer, E., Singh, J.V., 1996. Organizational evolution, learning, and selection: A genetic-algorithm-based model. Academy of Management Journal 39 (5), 1322–1349.

de la Maza, M., Yuret, D., 1995. A model of stock market participants. In: Biethahn, J., Nissen, N. (Eds.), Evolutionary Algorithms in Management Applications. Springer, New York, pp. 290–302.

Dennis, A.R., George, J.F., Jessup, L.M., Nunamaker Jr., J.F., Vogel, D.R., 1995. Information technology to support electronic meetings. MIS Quarterly 12 (4), 591–624.

DeSanctis, G., Poole, M.S., Lewis, H., Desharnais, G., 1992. Using computing in quality team meetings: Initial observations from the IRS-Minnesota project. Journal of Management Information Systems 8 (3), 7–26.

Gallupe, R.B., DeSanctis, G., 1988. Computer-based support for group problem-finding: An experimental investigation. MIS Quarterly 12 (2), 277–296.

Gillenwater, E.L., Conlon, S., Hwang, C., 1995. Distributed manufacturing support systems: The integration of distributed group support systems with manufacturing support systems. Omega 23 (6), 653.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Grohowski, R., McGoff, C., Vogel, D., Martz, B., Nunamaker, J., 1995. Implementing electronic meeting systems at IBM: Lessons learned and success factors. MIS Quarterly 14 (4), 369–383.

Lam, S.K., 1997. The effects of group decision support systems and task structures on group communication and decision quality. Journal of Management Information Systems 13 (4), 193–215.

Marks, R.E., Midgley, D.F., Cooper, L.G., 1995. Adaptive behaviour in an oligopoly. In: Biethahn, J., Nissen, N. (Eds.), Evolutionary Algorithms in Management Applications, Springer, New York, pp. 225–239.

McGrath, J.E., 1984. Groups: Interaction and Performance. Prentice-Hall, Englewood Cliffs, NJ.

Mood, A.M., 1950. Introduction to the Theory of Statistics. McGraw-Hill, New York.

Nunamaker, J.F., Dennis, A.R., Valacich, J.S., Vogel, D.R., George, J.F., 1991. Electronic meeting systems to support group work. Communications of the ACM 34 (7), 40–61.

Rees, J., Koehler, G.J., 1999. A new direction in group support system theory: An evolutionary approach to group decision-making, in review.

Sikora, R., Shaw, M.J., 1996. A computational study of distributed rule learning. Information Systems Research 7 (2), 189–197.

Simon, H.A., 1977. The New Science of Management Decision. Prentice-Hall, Englewood Cliffs, NJ.

Vose, M.D., 1999. The Simple Genetic Algorithm: Foundations and Theory. MIT Press, Cambridge, MA.

Vose, M.D., Wright, A.H., 1995. Simple genetic algorithms with linear fitness. Evolutionary Computation 2 (4), 347–368.

Zigurs, I., Buckland, B.K., 1998. A theory of task/technology fit and group support systems effectiveness. MIS Quarterly 22 (3), 313–334.