LAMP-TR-045 June 2000
UMIACS-TR-2000-38
CS-TR-4147

Quantifying and Interpreting the Effect of Intelligent Information Exchange Between Chromosomes in a Human Simulation of a Genetic Algorithm

Terry P. Riopka, Mona Diab, Peter Bock

Language and Media Processing Labratory
Instititue for Advanced Computer Studies
College Park, MD 20742

Abstract

A genetic algorithm is simulated using human beings as "chromosomes" in a preliminary study intended to quantify and interpret the effect of intelligent information exchange on genetic algorithm performance. Two factors are varied: the amount of information supplied to the cohort and the type of data manipulation allowed during the exchange. A human simulated genetic algorithm is run for each combination of factors as well as a machine simulation for comparison. Qualitative analysis of recorded conversations indicate extensive use of memory and development of block biases during genetic algorithm evolution. Informal analysis shows that genetic algorithm simulations using complex data manipulations combined with exact knowledge of string fitnesses seem to out-perform a standard machine implementation for the given optimization fitness function. Interestingly, polar combinations: simple data manipulation/minimum information and complex data manipulation/maximum information simulations seem to out-perform other combinations.

***The support of the LAMP Technical Report Series and the partial support of this research by the National Science Foundation under grant EIA0130422 and the Department of Defense under contract MDA9049-C6-1250 is gratefully acknowledged.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2000		**00-00-2000 to 00-00-2000**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Quantifying and Interpreting the Effect of Intelligent Informaiton Exchange between Chromosomes in a Human Simulation of a Genetic Algorithm	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Maryland,Institute for Advanced Computer Studies,Language and Media Processing Laboratory,College Park,MD,20742**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**11**	

Quantifying and Interpreting the Effect of Intelligent Information Exchange Between Chromosomes in a Human Simulation of a Genetic Algorithm*

Terry P. Riopka
Dept. of Computer Science
The George Washington University
Washington, DC 20052
riopka@seas.gwu.edu

Mona Diab
Dept. of Linguistics and UMIACS
University of Maryland
College Park, MD 20742
mdiab@umiacs.umd.edu

Peter Bock
Dept. of Computer Science
The George Washington University
Washington, DC 20052
pbock@seas.gwu.edu

Abstract: A genetic algorithm is simulated using human beings as "chromosomes" in a preliminary study intended to quantify and interpret the effect of intelligent information exchange on genetic algorithm performance. Two factors are varied: the amount of information supplied to the cohort and the type of data manipulation allowed during the exchange. A human simulated genetic algorithm is run for each combination of factors as well as a machine simulation for comparison. Qualitative analysis of recorded conversations indicate extensive use of memory and development of block biases during genetic algorithm evolution. Informal analysis shows that genetic algorithm simulations using complex data manipulations combined with exact knowledge of string fitnesses seem to out-perform a standard machine implementation for the given optimization fitness function. Interestingly, polar combinations: simple data manipulation/minimum information and complex data manipulation/maximum information simulations seem to out-perform other combinations.

Keywords: genetic algorithm, human simulation, intelligent recombination

1. Introduction

A genetic algorithm can be interpreted as a process of simulated evolution in which *chromosome* strings gradually modify the information encoded in their schemata. Some form of replication based on string fitness is typically used to simulate natural selection and incorporate the Darwinian notion of survival of the fittest. In addition, strings are altered according to the probabilistic rules of various operators, of which crossovers and mutations are the most well known. In this *information exchange* phase, the strings (heretofore referred to as *chromosomes*) exchange symbols probabilistically to try to improve their fitness in order to survive through to the following generation. A *building block hypothesis* has been suggested but not theoretically proven to be the primary impetus for solution improvement over generations. Efforts to develop analytical methods to determine the efficacy of problem coding/operator combinations have met with limited success but, nevertheless, strongly support this contention. According to the hypothesis, chromosomes in the genetic population exchange varying sized blocks of consecutive symbols which, over time, tend to proliferate in the components of chromosomes that survive from one generation to the next. In current implementations of genetic algorithms, the application of the various operators is probabilistic. In most cases these probabilities are chosen based on empirical studies and heuristics and are often non-optimal for any given application.

An alternative mechanism is proposed which uses memory and logical reasoning to determine how information is exchanged between chromosomes. If memory, logical reasoning and more complex data manipulation could be exploited to streamline the search for these building blocks, the efficacy of the genetic algorithm search might be improved and could also lead to more robust automated genetic algorithm

In Proc. of the 6th International Conference on Artificial Intelligence Applications, Cairo, Egypt, 1998.

search. To investigate this, a preliminary study was conducted using a human simulation of a genetic algorithm, under the assumption that human beings are capable of intelligent information exchange involving logical reasoning, memory and various forms of data manipulation. Thus, instead of relying on the probabilistic application of crossovers and mutations for information exchange between chromosomes, human beings are used to decide which information and how information is exchanged. Therefore, the objective of this research is to simulate a genetic algorithm using human beings and to quantify and interpret the effect of intelligent information exchange on genetic algorithm performance.

2. Human Genetic Algorithm Simulation

To simulate a genetic algorithm using human subjects, a standard genetic algorithm (SGA) was implemented on a computer using an algorithm whose cycle of evolution was made to interface with human subjects in the following manner. Each human subject (one for each string in the population and hereafter referred to as a humasome) interacts with a personalized interactive program that leads the cohort through the evolutionary cycles of a genetic algorithm simulation. Following the standard implementation model for genetic algorithms, initial random strings are evaluated and assigned to each humasome. Humasomes are then randomly mated, and asked to meet with their mate to discuss their two strings. Following a brief discussion, symbols may or may not be exchanged. The humasomes then return to their interactive programs and enter their new strings. A mutation operator is then applied to the input strings, after which the strings are reproduced in proportion to their fitness, evaluated and redistributed to the humasomes. This cycle is then repeated for the desired number of generations.

Various parameters affect the performance of a genetic algorithm in any given application, including the application itself, or fitness function in the context of the genetic algorithm. Additional parameters are: the number of strings in the population, the size of the strings, the number of symbols in the coding, the types of operators and the probabilities associated with them. In the case of the human genetic algorithm (HGA) simulation, information exchange was determined not by fixed probabilities, but by the decisions made by the humasomes. Two parameters were postulated to affect the decisions of the humasomes: amount of information supplied to the humasomes regarding the fitness of their strings, and the type of data manipulation allowed during an exchange of symbols. The latter two parameters were factors of the experiment and all others were fixed.

An optimizing function of moderate difficulty was selected based on the results of a HGA pre-pilot study which suggested that reasonable solution convergence could occur within about 10 generations of evolution. It was important that the cohort did not immediately guess the solution to insure interesting results, nor take too long to arrive at a reasonably good solution so that differences in convergence time (as factors were varied) could be observed. Therefore, the following fitness function was selected, requiring a correct sequencing of a set of symbols to optimize the result:

$$f(s_i) = b + q + 0.25 \bullet p \qquad [1]$$

where s_i is the string to be evaluated, b is the number of symbols in a correct position, q is the number of correct pair subsequences, and p is the number of pairs of duplicate symbols.

A string length of five was chosen to make it easier for the humasomes to remember past strings, thus encouraging the use of memory and past experience. A six symbol alphabet was chosen to insure that humasomes would not guess the optimum string with high confidence (computed to be greater than 92%). Vowels were excluded from the potential set of symbols to prevent humasomes from developing word biases. Numbers were excluded to prevent humasomes from assuming mathematical relationships between symbols. Favourite and least favourite symbols were excluded to reduce the bias humasomes may have

toward certain symbols. Finally, a random mapping function was used to map symbols to numbers, blinding both the controller and humasomes, to reduce possible bias due to alphabetical ordering of characters.

3. HGA Experiment

In order to quantify and interpret the effect of intelligent information exchange on genetic algorithm performance, a series of experiments was designed to validate the following research hypotheses:

Research Hypothesis 1:

Performance of an HGA will vary with the type of data manipulation and the amount of information supplied during information exchange for the given fitness function.

Research Hypothesis 2:

The extent to which humasomes make use of memory and logical reasoning will vary with the type of data manipulation and the amount of information supplied during information exchange for the given fitness function.

For purposes of comparison, an SGA control was tested using the same implementation model and same fixed parameters, but using a crossover operator for information exchange. Mutation rates for all HGA and SGA experiments were machine controlled and fixed.

3.1 Experiment Factors

Two different levels for the amount of information supplied to the humasomes prior to information exchange were tested. *Maximum information* provided the humasomes with the absolute fitness value of their string as computed by the fitness function. *Minimum information* provided the humasomes only with information regarding string survival.

Two types of data manipulation were tested. *Standard data manipulation* allowed humasomes to exchange symbols using two-point crossovers only. *Complex data manipulation* allowed humasomes to exchange symbols in any manner desired.

3.2 Performance Metrics

To measure the effect of the factors on HGA performance, two metrics were used. The first relates to how close the strings of a population are to the optimum fitness: the *Convergence Measure* (CM), given for generation j by:

$$CM_j = \frac{\frac{1}{n}\sum_{i=1}^{n} f(s_i)\Big|_{\text{generation}=j}}{F_{max}} \qquad [2]$$

where s_i is the i^{th} string of a total of n strings in generation j and F_{max} is the maximum possible fitness using the given fitness function f. A second metric for measuring HGA performance, the *Performance Matching Generation Count*, was the number of generations required for the SGA to match the convergence measure of the HGA in the last generation of the HGA simulation.

To measure the extent to which humasomes make use of memory and logical reasoning, three metrics were used: the number of explicitly stated explanations regarding motivations for information exchange (humasome reasoning), the number of explicitly stated references to previous strings (use of memory) and the number of explicitly stated symbol combination biases.

3.3 Experiment Procedure

HGA Simulation. For each HGA simulation, a cohort was selected consisting of 10 college level students from varying technical educational backgrounds, a variety of nationalities, with approximately equal proportions of each gender. Subjects were seated in a closed, air-conditioned room in front of numbered computer terminals as shown in figure 1. The interface program, written in ANSI C, ran on a UNIX mainframe and was accessed by each subject through a Macintosh terminal. A super-program, operated by the controller, was used to coordinate inter-program communication between the interface programs and control the flow of the experiment. The experiment proceeded as follows. After seating the cohort, a brief oral explanation of the cycle of evolution was presented, *excluding* a detailed explanation of fitness functions. Participants were told to treat the experiment like a game, where the goal was to identify the best set of symbols, given the conditions of the experiment. In addition, an instruction sheet was provided describing how to initiate the interface program, how and when to use the sound recorders, and how symbols were to be exchanged. Approximately 5 minutes were allotted for questions to clarify the rules of the experiment. During the experiment, only questions regarding a clarification of the rules of information exchange were answered by the controllers, and discussions were allowed only between humasome mates.

First, a brief pretest was administered through each interface program to determine favourite and least favourite letters for each humasome. Results were used to modify the HGA alphabet on the fly to reduce bias. The cycle of evolution proceeded as follows. Initial strings were randomly generated by the super-program and then randomly distributed to each humasome, displaying each humasome's string on the appropriate monitor. In the case of minimum information, the humasome was told whether or not the string had survived from the previous generation (except in the very beginning when the strings are random). In the case of maximum information, a fitness value also accompanied the display of the string. At this time, each humasome was also informed of their mate by number (randomly determined by the computer). Humasomes were then instructed to meet with their mates, start recorders and begin discussing their strings with the intent of possibly (but not necessarily) exchanging symbols in a manner consistent with their initial instructions. Paper and pencil were provided to encourage participants to *remember* past strings and past string performance. A time limit of approximately 3 minutes was set for information exchange with a suggestion to "hurry" issued 30 seconds before and every 30 seconds thereafter until the humasomes complied of their own free will. Once information exchange was completed, humasomes were instructed to stop the recorders, return to their terminals and to input their newly modified strings into their interface program.

A mutation operator was applied by the super-program using a mutation rate of 0.02 (2 mutations per 100 symbols processed). Note that this same mutation rate was used in the SGA machine implementation later for comparison. Strings were then reproduced in proportion to their fitness, evaluated and redistributed to the humasomes. If a humasome's string survived, that string was funneled to that humasome; all others were randomly redistributed to the remaining participants. Once redistributed, the strings were displayed on the appropriate monitors with the corresponding information regarding string fitness, and the entire cycle repeated. Ten generations were run, for a total duration of approximately 45 minutes (with no breaks) for each simulation.

SGA Control. As a control, 30 repetitions of an SGA machine simulation were run with different random seeds, using the same mutation rate as for the HGA (0.02) with three different crossover probabilities: 0.60, 0.75 and 0.85. The best run out of the 30 simulations for each crossover rate was used in comparisons with the HGA. The same code was used for both HGA and SGA simulations with the only difference being the application of the crossover (which, in the HGA, were substituted by human interaction).

4. Results

Fitness values for all strings after each generation (for both HGA and SGA simulations) were stored to data files for later analysis. Recorded conversations were codified using a list of acronyms shown in figure 2 which were then entered into a spreadsheet for analysis. Approximately 6 hours per experiment were needed to properly codify the conversations. Acronyms are related to 1. humasome reasoning expressed during information exchange, and 2. general observations regarding humasome behavior.

HGA experiments are labeled as follows in all figures: StdMinInfo for standard data manipulation and minimum information, StdMaxInfo for standard data manipulation and maximum information, CplxMinInfo for complex data manipulation and minimum information, CplxMaxInfo for complex data manipulation and maximum information. The label StdNoInfo is used for the SGA simulation, since the machine simulation can be considered to be equivalent to a HGA experiment where the humasomes are given no information.

Convergence Measure (CM) was used as a measure of the GA performance. The best SGA result was used for comparison in Table 1. The following table lists confidences by which the CMs in the last generation of the simulations (generation #9) referred to as "A" simulations are greater than those referred to as "B" simulations. The confidences are calculated based on a t-test using 9 dof (number of strings per generation - 1).

A \ B	StdNoInfo	StdMinInfo	StdMaxInfo	CplxMinInfo	CplxMaxInfo
StdNoInfo	*	45	59	52	37
StdMinInfo	55	*	59	56	49
StdMaxInfo	42	41	*	46	8
CplxMinInfo	48	44	54	*	33
CplxMaxInfo	63	51	92	67	*

Table 1: Confidence that the CM for simulation A is greater than the CM for simulation B.

The standard deviations across generations were calculated to compare the performance of the HGA simulation with the SGA simulation.

The number of generations required for the best machine SGA to match the HGA simulation performance was also plotted (not shown). Negative values indicated that the SGA converged that many generations sooner than the HGA. Positive values indicated that the SGA took that many generations longer to converge to the equivalent HGA's last generation's average string fitness. Counts of the block biases developed by the humasomes in the HGA simulation simulations were also tracked across generations and plotted (not shown).

The number of explicit memory references verbally expressed by the humasomes during each of the HGA simulations were counted. Table 2 illustrates the confidences by which the counts are significantly different from one another.

Twelve reasoning codes were extracted from the transcription of the recorded conversations between humasomes. The frequency of these occurrences were plotted per simulation in one composite chart for comparison (not shown).

	StdMinInfo	StdMaxInfo	CplxMinInfo	CplxMaxInfo
StdMinInfo	*	98	99	99
StdMaxInfo	98	*	98	95
CplxMinInfo	99	98	*	70
CplxMaxInfo	99	95	70	*

Table 2: Confidences that the number of explicit memory references is different from one simulation to another.

5. Conclusions and Recommendations

Obviously, the small sample in this preliminary study precludes any firm conclusions. Nevertheless, a number of interesting qualitative observations can be made.

SGA performance was found to be similar over a range of crossover probabilities and across generations for this fitness function. As a result, only the best SGA simulation is included for comparison purposes, simplifying the comparison with HGA experiments.

From table 2 there is a 92% probability that the simulation CplxMaxInfo (complex manipulation combined with maximum information) results in performance that is greater than the results obtained from the standard manipulations with maximum information (StdMaxInfo). This was supported by qualitative observations of the cohort in which some indicated frustration at not being able to test certain hypotheses due to the constraint on data manipulation.

The result that StdMinInfo and CplxMaxInfo simulations seem to perform better than the other HGA simulations (as well as the machine simulation StdNoInfo) is also supported by the number of generations required by the machine to catch up to the HGA performance. Note that the best machine simulation out of a set of 30 Monte Carlo'd runs was used for comparison. In both cases the machine simulation needed in excess of 21 generations to arrive at the same performance level reached by the StdMinInfo and CplxMaxInfo simulations. Results suggest that the machine was able to reach the performance level achieved by the other two simulations in generation 9, in less than 4 generations. These results point to an interesting observation: when a genetic algorithm simulation is bound by standard GA data manipulations and is given minimal information, its performance is better than when the same GA is given either no information at all or maximum information. Note, its performance is comparable to (but not quite as good as) the HGA simulation bounded by complex manipulations and given maximum information.

Unfortunately, there is only a 63% possibility that the apparent best experiment (CplxMaxInfo) is better in performance than the machine simulated genetic algorithm. And, in fact, the standard manipulation with minimum information (StdMinInfo) appears to be (inexplicably) a close second best at 55% (see table 2). It may be that to make full use of the amount of information, complex data manipulation is required. Otherwise, the amount of information is not as important and simple manipulations do just as well.

By looking at the Block Biases developed across generations per simulation, all except the StdMinInfo simulation seem to show an increase near the middle generations of evolution. This is consistent with the idea that people might begin with few biases, then develop biases as more information becomes available to them, and finally reduce their biases as they converge to a possible solution.

Results show that the number of memory references in the StdMinInfo simulation is greater than the number of the same type with more than a 98% confidence level (using a single-tailed version of the data in table 2). This suggests that in cases where data manipulation was constrained with minimal information, more references to previous strings occurred, that is, people tended to rely more on their memories to try and extract as much information as possible in a difficult situation. However, in the case of complex data manipulation, this contrast is not as great (only a 70% confidence in a difference).

Regarding reasoning codes, it is interesting to note the diversity of reasoning in such a short simulation. Symbol homogeneity and heterogeneity (in the authors' opinion, the simplest of all reasoning) were most often cited in all simulations except in CplxMaxInfo where Symbol Location Reasoning clearly prevailed. This suggests that given a large amount of information and complex manipulations, people were able to progress to a more complex hypothesis (and in the case of this fitness function, more relevant) more quickly.

In general, it seems feasible that memory and reasoning could augment the search for potential solutions in a genetic algorithm, especially if a larger population were used (as is the case in most GA simulations). A learning automaton could be a very qualified substitute for the humasome and could provide a very testable machine simulation for exploring further hypotheses. For example, in the human experiment, one important characteristic of the program was to funnel strings that survived to those humasomes that were responsible for that string's survival. In this way, the humasomes could develop consistent hypotheses as to the reason for their string's continuing success. Given this same strategy, a learning automaton might be able to develop specific biases similar to those in a human by identifying the building blocks consistent with the string's survival and fitness. The building block hypothesis would then no longer be a hypothesis, but a real mechanism exploited by the genetic algorithm for solution search. How this affects the search for global optimal solutions and the effect on general convergence rate would have to be determined. However, the results of this experiment perhaps remotely suggest that such an investigation might be warranted.

Acknowledgements

The second author would like to acknowledge DARPA/ITO Contract N66001-97-C-8540 for travel funding support.

6. References

1. Goldberg, D. E. (1989), Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley.

2. Holland, J. H. (1987) "Genetic Algorithms and Classifier Systems: Foundations and Future Directions". Genetic Algorithms and Their Applications: Proceedings of the 2nd Int. Conf. on Genetic Algorithms, pp. 82-89.

3. Mitchell, M. (1996). An Introduction to Genetic Algorithms, The MIT Press.

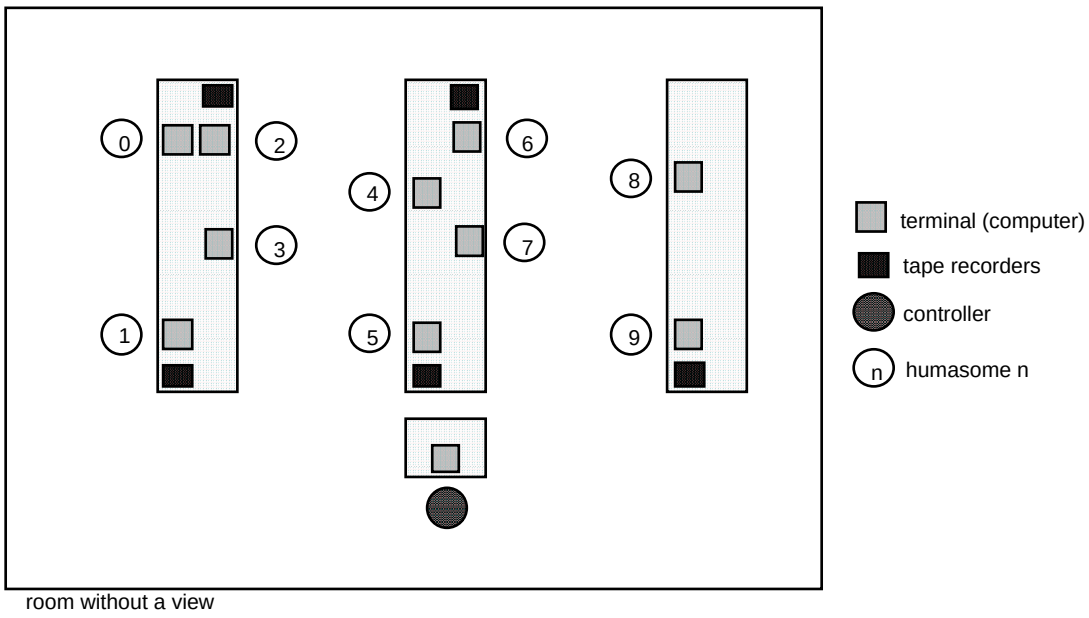

Figure 1: Layout schematic of the laboratory.

Reasoning Code

ALTSR:	ALTernating Symbol Reasoning
AR:	Ambiguous Reasoning
CSR:	Cancelling Symbol Reasoning
ER:	Emotional Reasoning
HESR:	HEterogeneous Symbol Reasoning
HOSR:	HOmogeneous Symbol Reasoning
HRSR:	HigheR Symbol Reasoning
GDR:	Generation Dependence Reasoning
LIMSR:	Limiting pool of Strings Reasoning
MLR:	Meta-Level Reasoning
SLR:	Symbol Location Reasoning
SNLR:	Symbol Not Location Reasoning
SOR:	Symbol Order Reasoning
SSSR:	Separate Similar Symbol Reasoning
SYR:	SYmmetry Reasoning

Observation Codes

BOC(n):	Backward Other Context (level)
BSC(n):	Backward Self Context (level)
DBB:	Developed Block Bias
DNC:	Do Not Care
DSB:	Developed Symbol Bias
IE:	Information Exchange
NIE:	No Information Exchange
NSM:	No Symbol Manipulation
OFR:	Other Fitness Reflection
SFR:	Self Fitness Reflection
SM:	Symbol Manipulation

Figure 2: List and brief description of acronyms used to codify recorded human conversations.

Filename: HGA_Paper.rtf
Directory: F:\denise\amy\treports\MONA\MONA
Template: C:\Program Files\Microsoft Office\Office\Normal.dot
Title: Quantifying and Interpreting the Effect of Intelligent Information Exchange Between Chromosomes in a Human Simulation of a Genetic Algorithm
Subject:
Author: Terry P. Riopka
Keywords:
Comments:
Creation Date: 6/8/00 11:49 PM
Change Number: 2
Last Saved On: 6/8/00 11:49 PM
Last Saved By: mona diab
Total Editing Time: 1 Minute
Last Printed On: 6/15/00 5:35 PM
As of Last Complete Printing
Number of Pages: 9
Number of Words: 3,410 (approx.)
Number of Characters: 19,442 (approx.)