AIAA 2000-4844

Experience with a Genetic Algorithm Implemented on a Multiprocessor Computer

G. Plassman and J. Sobieszczanski-Sobieski
NASA Langley
Hampton, VA

8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization
6-8 September 2000
Long Beach, California

AIAA-2000-4844

EXPERIENCE WITH A GENETIC ALGORITHM IMPLEMENTED ON A MULTIPROCESSOR COMPUTER

Gerald E. Plassman
Computer Sciences Corporation
3217 North Armistead Avenue
Hampton, VA 23666-1379
757 766 8226, fx-2571
g.e.plassman@larc.nasa.gov

Jaroslaw Sobieszczanski-Sobieski,
Manager, Computational AeroSciences
corresponding author,
NASA Langley Research Center, M/S 139
100 NASA Way
Hampton, VA 23681,
757 864 2799, fx-9715
j.sobieski@larc.nasa.gov

Abstract

Numerical experiments were conducted to find out the extent to which a Genetic Algorithm (GA) may benefit from a multiprocessor implementation, considering, on one hand, that analyses of individual designs in a population are independent of each other so that they may be executed concurrently on separate processors, and, on the other hand, that there are some operations in a GA that cannot be so distributed. The algorithm experimented with was based on a gaussian distribution rather than bit exchange in the GA reproductive mechanism, and the test case was a hub frame structure of up to 1080 design variables. The experimentation engaging up to 128 processors confirmed expectations of radical elapsed time reductions comparing to a conventional single processor implementation. It also demonstrated that the time spent in the non-distributable parts of the algorithm and the attendant cross-processor communication may have a very detrimental effect on the efficient utilization of the multiprocessor machine and on the number of processors that can be used effectively in a concurrent manner. Three techniques were devised and tested to mitigate that effect, resulting in efficiency increasing to exceed 99 percent.

Notation

Test Case Parameters:

N	Number of processors
CF	Generations between global communication
ND	Number of communication domain
NDV	Dimension of design space
NPOP	Global population
NGEN	Generation limit

Computed Robustness Metrics:

V	Volume of best feasible solution (X) found
U	Benchmark (N=1) normalized V
B	Generation containing X
D	Normalized standard deviation of B population volumes
R	Mean radius of B population design point distribution

Computed Scalability Metrics:

TTe	Elapsed time to complete all BCB generations
A	Problem analysis portion of TTe
O	BCB computation portion of TTe
M	BCB communication portion of TTe
%A	Percent A of TTe
%O	Percent O of TTe

- %M Percent M of TTe
- S Elapsed time based speedup factor (over N=1 case)
- E Efficiency in percent (100*(S/N))

Computed Termination Metrics:

- TTe' Elapsed time in percent of TTe with no termination criterion
- E' Efficiency in percent of E with no termination criterion
- G Final or termination criterion satisfying generation

Additional Symbolic Metrics:

- T Total computation time on a single processor
- Tp Portion of T which can be distributed
- Tn Portion of T which cannot be distributed
- Tc Total communication time
- Rt The ratio TTe/T, the reciprocal of S
- Ts The total time saved, T - TTe

Introduction

Genetic Algorithm (GA), and its variant known as Evolutionary Algorithm (EA), are optimization techniques that appear to be ideal candidates for efficient implementation on a machine with many processors (physical or virtual). This is so because GA and EA are based on generation and evaluation of a population of candidate designs, each of which may be analyzed independently. However, in addition to the above independent analyses, the algorithm requires a certain amount of cross-communication and other operations that cannot be distributed. Therefore, the scalability is not ideally linear.

This study purpose was to determine experimentally the GA scalability and to assess three techniques devised for improving that scalability. Robustness of solution quality and termination criteria are also addressed.

The Algorithm

The algorithm known as the Bell-Curve Based Genetic Algorithm (BCB) introduced in ref. 1, is used. Its performance was further elaborated on in ref. 2. BCB may be categorized as the Evolutionary Algorithm because unlike the conventional GA it does not use a binary-string based reproduction and mutation mechanism. Instead it employs a Gaussian distribution to generate a child from a pair of parents.

To be more specific, the algorithm step-by-step recipe is as follows:

1) Generate a population of designs by any technique commonly used in a conventional GA.
2) Analyze each design for the value of the objective function and constraints. For each design generate a single number "measure of fitness" combining the value of the objective (the smaller the better) and of the constraints (negative = satisfied, zero = active (critical), positive = violated); which means that the smaller the better).
3) Pair-up the designs to form parents for mating; rewarding fitness with more chances to mate. BCB uses the electronic roulette to do this, similar to the conventional GA.
4) Generate a child. The distinguishing features of the BCB algorithm are concentrated in this step, usually referred to as the crossover in the GA terminology. Therefore, the step is described in more detail:
    - Consider a design space in n-dimensions. Design points P1 and P2 are the parents. The hyperline P1-P2 connects the parents and extends beyond P1 and P2 to infinity.
    - Parents P1 and P2 generate a child at C. Point C is at the end of radius r emanating from B. Point B is located on the hyperline P1-P2 by chance governed by a probability distribution defined by the "bell curve" peaking at point M; the highest probability is in B falling at M. There is also a small chance for B falling outside of the segment (P1, P2). Point M may be the mid-point of the (P1, P2) segment, or its position may be shifted toward the parent of higher fitness.
    - The radius "r" defines the distance from point B to the surface of an n-1 dimensional hypersphere orthogonal to the segment (P1, P2) and centered at the point B. The radius length is governed by another "bell curve" centered on zero. A point to represent a child design is generated on the surface of this hypersphere by a uniform probability distribution.
    - The X coordinates of the child are checked against the side constraints and reset accordingly.
5) Repeat steps 3 to 4 to produce the entire offspring generation that tends to be better fit than its parent generation. Since the X' s measure the

design characteristics directly, there is no need for binary string manipulation.

6) Group together the newly generated children population and the previous population from which their parents were drawn and select the most fit individuals. These individuals represent the next generation. The size of the generations is kept the same by discarding the least fit half of the grouped populations. Then repeat from step 3 until the maximum number of generations or satisfaction of some alternate termination criterion is reached.

No random mutations are used with the above process because they already occur by virtue of the hyperline extending beyond P1 and P2 and the orthogonal hypersphere feature.

Efficiency of Coarse-Grained Operation Engaging Many Processors Concurrently

In general, such operation will have a computational part that can be distributed, a computational part that cannot be distributed, and a certain amount of communication that tends to be a function of the number of processors engaged in the distributable part of the computation. In our paper "communication" always refers to inter-processor communication. Our test problem is small enough so that no processor-to-mass storage communication is required. If that were not the case, the processor-to-mass storage communication time would become a factor.

Suppose that the total elapsed time, T, of the computation at hand when executed on a single processor machine consists of the sum

(1) $T = Tp + Tn$

where Tp is the sum of the elapsed times of the parts that can be distributed (parallelized), and Tn is the analogous sum for the non-distributable parts.

Let us now assume that the above computation is executed on N concurrently operating processors on a machine whose total number of processors $NP \geq N$. We will also recognize that even though the processors operate independently, they may need to communicate with each other as required by the solution algorithm. The communication time is, usually, a function of N, $Tc = Tc(N)$. Then, TTe, the total elapsed time to solution is:

(2) $TTe = Tp/N + Tn + Tc(N)$

A plot of TTe and its components Tp/N, Tn, and Tc, assuming linearity of the latter, is shown in Figure 1. The plots indicate the importance of Tn and Tc as limitations on N that can be used effectively.

Comparing TTe to T, we have a few metrics of obvious interest. The first metric is the ratio of TTe/T, denoted Rt, which measures, relative to T, the time saved by using a multiprocessor computer. It can be expressed as

(3) $Rt = TTe/T$
$= (1/N)(1 + N\,Tn/Tp + N\,Tc/Tp)/(1 + Tn/Tp)$

The inverse of Rt measures the speed-up

(4) $S = 1/Rt$
$= N(1 + Tn/Tp)/(1 + N\,Tn/Tp + N\,Tc/Tp)$

The total time saved, Ts, is

(5) $Ts = T - TTe$
$= Tp(1 - 1/N - Tc/Tp)$

Finally, the ratio of S/N, denoted E, that measures the efficiency of utilization of a set of N processors becomes

(6) $E = S/N$
$= (1 + Tn/Tp)/(1 + N\,Tn/Tp + N\,Tc/Tp)$

The metrics TTe and Ts are of a primary interest to the machine user, while the metrics S and E matter most for the machine operator.

The limits of S and E are instructive. Assume L and s to be large and small numbers, then

(7) Limit S for $N \rightarrow L$
$= (1+Tn/Tp)/(Tn/Tp + Tc/Tp)$

(8) Limit S for $N \rightarrow L$ and $Tn/Tp \rightarrow s$ and $Tc/Tp \rightarrow s$
$= Tp/(Tn+Tc)$

(9) Limit E for $N \rightarrow L$
$= (1+Tn/Tp)/(N(Tn/Tp + Tc/Tp)) \rightarrow 0$

To illustrate the orders of magnitude, suppose that Tn/Tp = .01 and Tc/Tp = .01. Then, for N = 100 and N = 1,000,000, we have Table 1.

Table 1. Sample Parallel Computing Metrics

N	100	1,000,000
TTe	Tp*0.03	Tp*0.020001
Rt	0.0297	0.0198
S	33.7	50.5
Ts	Tp*0.98	Tp*0.99
E	0.337	0.0000505

It is obvious that we need to keep Tn/Tp and Tc/Tp as low as possible in order to achieve high S and E, and low Rt. It is apparent that increase of N by four orders of magnitude produces very little improvement in terms of Rt, S, and Ts in the presence of even quite small Tn/Tp and Tc/Tp. These parameters depress significantly S and E, and they severely limit the number of processors that can be effectively used. Indeed, eq. 6 suggests that for Tn/Tp = 0.01 there is little incentive to increase N beyond 100, even if Tc/Tp = 0. For that case eq. 4 indicates the asymptotic limit on speedup, S, is equal to the inverse of the fraction of total cost which is serial, a characteristic which became known in the trade as Amdahl's Law (ref. 3). This does not negate the positive fact expressed in the Ts metric, one of primary interest to the user, that the time saving nearly equals Tp for N = 100.

One should note that in Table 1, the ratio Tc/Tp is assumed constant. Should it be increasing proportionally to N, as would be likely if a global communication were maintained, TTe would have reached 1.02*Tp = T + Tn > T for N = 100. The multiprocessor machine advantage would evaporate! This example clearly shows that one must be extremely wary of introducing a processor-to-processor communication when implementing an algorithm on a multiprocessor machine.

Implementation on Many Processors That May Operate Concurrently - Three Techniques

Cross-communication among the processors is a significant contributor to Tc affecting the metrics introduced in the preceding section. Three techniques have been devised to keep the ratios Tn/Tp and Tc/Tp as low as possible. Basically, in a Genetic Algorithm the analysis time of each individual design falls into the Tp category, while the time for scanning the population to form the parents and children is in the Tn category, and that operation generates also Tc. Attempting to reduce Tn and Tc, we partition the population of designs and break down the above operation into concurrent ones, each performed within the confines of the individual partitions. We have found, however, that although the communication limited in that manner does indeed reduce Tn and Tc, it also tends to degrade the quality of the final result, the constrained minimum is not as low as the benchmark. To mitigate that undesirable effect, a periodic communications across the partition boundaries are performed. Three different schemes for the inter-processor, periodically refreshed communication were tried as described below.

Figure 2 symbolically compares the distribution, over time and across processors, of principal computational and any necessary communication tasks for a serial and three parallel technique implementations. Vertical direction in each diagram in the figure corresponds to the flow of time, while horizontal direction in each diagram symbolizes distribution over the concurrently operating processors. Task types are identified by upper case letters, A for the combination of design point analysis and their fitness evaluation, G for all remaining BCB algorithm computations, and C for any communication related overhead. The diagrams shown are for N = 3, but they indicate patterns readily extendable to any N. The subscripts "p" and "n" used with G and C distinguish parallelizable from non-parallelizable parts of these operations. Note that task A computations are entirely independent of each other by definition.

Serial diagram, the single processor implementation, is a benchmark. Parallel 1, in the figure, represents a basic technique for parallel BCB implementation, limited to a partitioning of the population analysis. This allows parallel A tasks across processors, but leaves Gp tasks sequential on a single processor. A communication overhead, represented by task C, is a penalty for maintaining global solution convergence over this partitioning. That penalty increases with N as well as with the requested frequency of communication achieving a global population update. Communication frequency, CF, is specified in terms of the number of population generations between successive global updates. For a given communication frequency task C cost remains a function of N. While task C involves all processors, data are broadcast from and received by the single processor on which serial tasks are performed. Thus C is symbolized as a single processor task. For this technique, the elapsed time for parallel computations, To = Tp/N, is limited to the time of a single task A, while time Tn includes the sum of Gn and all Gp tasks. Task C requires Tc.

Parallel 2, in the figure, represents an extension to distribute tasks Gp across processors, effectively increasing "To" by the time of a single Gp task, while reducing Tn by the sum of all Gp tasks, resulting in a decrease in the ratios Tn/To and Tn/Tp. This distribution of Gp leads to an additional communication overhead task, Cp, similar to C in character. Presuming C and Cp costs do not dominate overall cost, this second technique for parallel BCB implementation results in a potential for dramatic improvement of S and E, representing significantly increased scalability when analysis cost is comparable with other BCB cost.

Parallel 3, in the figure, represents the third and final parallel BCB implementation technique. This technique focuses on reducing the elapsed time cost of communication overhead realized with Parallel 2, a cost of increased significance given the addition of Cp and reduction of elapsed Gp time. The reduction is achieved by a distribution of both C and Cp over multiple communication domain, represented by C' and Cp'. A related benefit is the associated distribution of Gn over communication domain, represented by Gn'. An additional motivation, given the increase in C and Cp with N, was the desire to benefit from the potential increase in scalability promised by the higher efficiencies, relative to Parallel 1, offered by employing larger numbers of processors.

While Parallel 2 utilizes a single base processor for performing Gn and managing C and Cp, Parallel 3 employs multiple base processors, one per communication domain, each concurrently addressing domain specific portions of Gn, C, and Cp. The implementation employed presumes a power-of-two number of processors, N, partitioned for any given communication task into a mutually exclusive two-dimensional covering set of ND equal sized domain. Consecutive communication tasks, occurring at CF specified generation intervals, are governed by a sequence of such domain sets. This sequence begins with a single global domain and continues with a number of multiple domain covering sets of varied granularity. Sequence length and pattern of granularity variation over element are both determined by N. A minimum granularity of four processors per domain is defined for all even numbered sequence elements. The odd numbered sequence elements define the use of power-of-four increases in domain size with corresponding power-of-two decreases in frequency, including a single (the first) sequence element specifying global communication. Iteration on this sequence is performed as necessary, with global domain usage forced for the final communication task. Such sequences define a multiple domain communication strategy favoring communication locality and concurrency, yet maintaining occasional communication over larger domain, including global communication.

Figure 2 illustrates the communication domain sequence applied for N = 64. This four-element sequence, notated as {1,16,4,16}, follows global communication with 16, 4, and 16 element multiple domain covering sets of 4, 16, and 4 processors per element, the product of covering set size, ND, and the common domain size being equal to N for each element. The illustrated cycle of communication domain readily extends for N equal to other powers of two. For example, the sequences {1, 32, 8, 32, 2, 32, 8, 32} and {1, 64, 16, 64, 4, 64, 16, 64} represent the order and size of covering sets to be addressed when N = 128 and N = 256. Notice that 8 is the minimum N for which Parallel 3 is distinct from Parallel 2.

The scalability of Parallel 1 is very limited by virtue of its serial G computation. Parent selection and child generation dominates G computation. While the former is global in character, the latter and more dominant of the two is independent over children and, given parent pair distribution, parallelizes like problem analysis with a communication requirement of similar character, leading to definition of Gp and Cp; the remainder of G becoming Gn. The resulting Parallel 2 version enjoys a dramatically extended scalability, particularly with frequent communication. Robustness is maintained. With Parallel 2 the dominant limitation on scalability is the growth of communication cost. Parallel 3, which reduces that cost, is most effective in extending scalability when N is large and communication is frequent (CF is low). Given the localized nature of most Parallel 3 communication events, maintaining robustness comparable with Parallel 2 may require a moderate reduction in CF, particularly with large N.

All BCB implementations employ the algorithm described in Section 2. The initial population is randomly distributed throughout the entire design space. Its definition is performed serially, based on a single random number seed, to ensure that distribution. Algorithm initialization includes a broadcast of the updated seed along with initial population distribution. Distinct processor-local initial populations ensure independent child generation based on the same seed. Total population, NPOP, values equal to an integer multiple of N ensure balanced processor workload. Feasible solution fitness is equal to objective value alone, while solution fitness at an infeasible design point equals the objective value plus a penalty. That penalty is equal to a fixed multiple of the maximum (most violated) constraint, where the multiple selected is on the order of expected objective value. Child generation on the surface of a hyper-sphere is performed in a normalized design space by coordinate scaling, a technique known to improve convergence.

Communication requirements are satisfied with Message Passing Interface (MPI) send/receive pairs. Domain base processors perform domain specific global (serial) computation and inter-processor message management on generations requiring inter-processor communication. Each base processor receives and assembles processor local parcels of domain population data prior to necessary domain specific global computation and redistributes updated parcels. Message volume is dominated by population coordinate data. Message count is reduced by concatenation of remaining population data, including volume, fitness, and maximum constraint, prior to communication. Parallel 1 requires two such communication exchanges, involving a total of 5 coordinate data sets, per affected generation, supporting both child generation and next generation population selection. Parallel 2 and Parallel 3 each require a third exchange, in support of parallel child generation. This additional exchange involves one communication of a doubled population coordinate data set, representing parent pairs, and a second standard coordinate data set, representing the resulting children population. This increases the total message volume per affected generation by about a factor of 8/5. Given ND communication domain, N processors, and a total population of NPOP, total message volume and count on associated generation are of order NDV*NPOP*(N-ND) and (N-ND), respectively, where NDV is the dimension of the design space. The reduced message count and total volume realized with increased ND, and associated increased concurrency and locality of their dispatch, identify the sources of communication cost savings provided by Parallel 3.

Points of intra-domain communication provide the opportunity to repartition domain populations over their constituent processors, with a corresponding potential for improved evolution of domain population prior to the next communication point. Two alternatives, one based on equal average fitness and a second based on design space locality, provided similar moderate improvement over no repartitioning. The equal fitness alternative was selected on the basis of its negligible overhead. The cost of extensive distance computations associated with the point locality alternative, even when mitigated by parallel techniques, is the source of its significant overhead.

The current implementation of child generation requires a per processor population (parent pool) of at least two when communication does not occur every generation (CF > 1), imposing a general limit on N = NPOP/2. Furthermore, robustness requires a maximum N = NPOP/4 under these conditions. These limitations identify NPOP as a critical factor for scalability as well as total computation cost. For a given problem size, excessive NPOP results in reduced algorithm effectiveness. An increase in optimal NPOP with problem size extends practical scalability limits with that size. While not tested, a modification of Parallel 3 to vary communication frequency with communication domain size, including CF = 1 on four-processor domain, would extend the limit of scalability with computational robustness to that allowing full analysis concurrency (N = NPOP) while managing communication overhead.

Test Case and Metrics

The hub frame structure described in ref. 4 is the test problem. Hub frame problems are composed of a two-dimensional arrangement of I-beams radiating from a common hub and rigidly attached to both hub and surrounding wall. They are optimized for minimum material volume under a concentrated force and moment loads applied at hub center, subject to constraints arising from limits on material stress, local and overall buckling resistance, and hub center displacement. Design space dimensionality and total constraint count depend on the number of the members while the dimensionality of the load-deflection equations remains constant, 3x3. Thus, the design space can be made large and rich in constraint nonlinearities, without expanding the analysis computational labor.

The 20-member hub frame problem of ref. 4, modified by replacement of a single hub center translational constraint with independent constraint of its two components, is the primary test problem. Table 2 summarizes member (MBR), design space dimension (NDV), and functional constraint (NCON) counts for this and other addressed problems. Problem size identifiers in the Results section consist of member count prefixed with "H". For example, "H20" identifies the 20-member problem.

Table 2. Hub Frame Size, Dimensionality, and Constraint Count

MBR	NDV	NCON
20	120	766
80	480	3046
180	1080	6846

Computed test metrics, defined in the Notation section, quantify robustness, scalability, and effectiveness of termination criteria. All termination criteria metrics are reported in the Results section, along with selected robustness and scalability metrics. Our definitions of robustness and scalability for this paper are presented in the following two paragraphs, along with identification of principal related metrics and characteristic summaries of metrics not reported in the Results section.

We define robustness as the ability to maintain solution quality, with respect to a serial benchmark, over increasing values of N and CF, i.e. larger processor arrays and reduced communication. Normalized solution volume, U, is considered to be the principal metric of robustness. Metrics D and R also quantify robustness in terms of the character of population convergence. These three are reported in the Results section. Metric B is sufficiently described as being generally within one percent of G. Metric V is equal to U times the benchmark V obtained on a single processor.

We define scalability as the ability to maintain high computational efficiency on processor arrays of increasing size, N, in terms of an ideal constant product of elapsed execution time and N, with minimal degradation in robustness. The principal metric of scalability, E, is an N normalized equivalent of S. Additional metrics identify sources of scalability and its degradation, and quantify their relative impact. Metric A consistently demonstrates a balanced distribution of the analysis workload to be a source of ideal speedup. Metrics O and M, the remaining components of TTe, respectively quantify the varying scalability of the three parallel algorithm implementations and the strength of their communication overhead. The metric %M is reported in the Results section along with E and O, since it identifies, more clearly than M, the impact of communication cost. The significant characteristic of communication cost M was its mild super-linear growth as a function of N over the range addressed. This implies %M will approach order N^2 growth while A and O computations remain dominant, underscoring the criticality of computational overhead.

The multiprocessor machine used in the testing was a Silicon Graphics Origin 2000 with 256 processors. Up to 128 processors was used. In order to experiment with different ratios Tn/Tp, the Tp magnitude was artificially increased by inserting in the test case analysis an "idle loop" computation that expanded Tp without affecting the mathematical model of the problem physics. Two levels of Tp expansion are tested. The first is based on a hub frame analysis with an artificial ten-fold increase in cost, designated by the problem identifier extension "/x10". The second, designated by the extension "/99p", is based on a hub frame analysis with an artificial cost increase of a magnitude such that the total analysis cost represents 99 percent of total execution time on a single processor. For example, test result for "H20/x10" and "H20/99p" identify 20-member hub frame analyses with these increased cost levels, distinguishing them from standard result identified by "H20/std". Result metrics independent of analysis cost, such as U; are identified with no extension.

Results

Results are obtained from a series of test sets, each providing scalability and robustness metrics for a given parallel technique and problem size/simulated complexity combination over power-of-two ranges of processors (N) and communication frequencies (CF). For each test set population size (NPOP) and generation limit (NGEN) are held constant. NPOP values used ensure balanced processor loading with equal power-of-two processor local populations. Test sets yielding scalability metrics ensure cross test standardization by employing no termination criteria other than a fixed limit, NGEN, on the total generation count for population evolution. Core test set results, establishing reported robustness and scalability, are based on the same initial seed value for distribution sampling. Additional testing, employing more effective sampling for parent selection and child generation, and providing averaged metrics over 15 random seeds substantiated these results. The remainder of this section provides summary comparison of parallel technique, primarily in terms of robustness metric N and scalability metric E, for the higher performance combinations of N and CF, when applied to various hub frame problems.

Table 3 demonstrates the poor scalability of Parallel 1 and dramatic improvement with Parallel 2, which parallelizes child generation, the dominant BCB computation. Improvement is greatest for large N and small CF. The data are for the ten-fold cost 20-member problem (H20/x10), a representation of the increased complexities of typical large scale engineering problems. Table 4 compares the BCB specific computation cost, O, of Parallel 2 with Parallel 1. Metric O quantifies the primary source of scalability degradation for Parallel 1, an implementation for which, on generations employing global communication, the represented computation is entirely serial. This characteristic is illustrated by the constant O value over increasing N for Parallel 1 when global communication is performed on every generation (CF = 1). Communication cost for Parallel 1 remained less than two and four percent for N = 32 and N = 64 for CF values greater than eight. Parallel 2 robustness compares well with Parallel 1, both demonstrating average solution degradation of less than four percent when processor local population is at least 4 ($N \leq 32$)

Table 3. Parallel 1 versus Parallel 2 Scalability for H20/x10 with NPOP=128 and NGEN=1000

Parallel 1: E over CF and N

CF	N=16	N=32	N=64
4	81.95	70.29	51.85
8	88.86	82.50	67.84
16	91.82	89.51	80.45
32	90.53	94.22	86.58
64	98.20	95.28	93.06

Parallel 2: E over CF and N

CF	N=16	N=32	N=64
4	98.45	94.53	78.63
8	100.03	96.88	88.33
16	100.63	98.45	94.02
32	98.28	97.17	97.12
64	99.02	101.16	98.31

Table 4. Parallel 1 versus Parallel 2 BCB Specific Cost in seconds for H20/x10 with NPOP=128 and NGEN=1000

Parallel 1: O over CF and N

CF	N=1	N=4	N=16	N=64
1	89.40	88.90	88.70	89.60
4	89.40	39.00	25.80	23.80
16	89.40	25.70	10.20	7.07
64	89.40	22.50	6.71	2.87

Parallel 2: O over CF and N

CF	N=1	N=4	N=16	N=64
1	82.30	26.10	12.10	9.07
4	82.30	22.20	7.15	4.08
16	82.30	21.00	5.67	1.99
64	82.30	20.70	5.31	1.44

Tables 5 and 6 compare Parallel 2 and Parallel 3 performance against H20 with NPOP increased to allow N = 128. Table 5 shows Parallel 3 robustness compares well with Parallel 2, both tolerating communication intervals as high as 32 generations, when processor local populations are at least four, with average solution degradation limited to about six percent. Table 6 provides additional Parallel 3 robustness metrics, normalized objective standard deviation (D) and mean radius of design point distribution (R) of generation NGEN population. Specific row and column of D each demonstrate a positive correlation with N and CF respectively, indicating population convergence to be slowed by both increased and prolonged fragmentation. A weaker form of this correlation extends to Parallel 3 U data in Table 5. Together these motivate the use of an upper bound on D as a termination criterion. While R demonstrates a similar behavior, and may represent a useful alternative termination criterion for problems with multiple dispersed local optima, it has the disadvantage of an increased overhead in the form of the required distance computations.

Table 5. Parallel 2 versus Parallel 3 Robustness for H20 with NPOP=256 and NGEN=1000

Parallel 2: U over CF and N — Parallel 3: U over CF and N

Benchmark (N=1) Volume (U=100) = 12800

CF	N=32	N=64	N=128
4	106.87	105.39	101.72
8	109.52	103.28	110.07
16	111.71	99.38	114.91
32	107.42	103.90	120.14
64	108.90	114.99	130.68

CF	N=32	N=64	N=128
4	109.13	106.64	108.82
8	105.15	99.45	108.51
16	106.32	103.04	111.40
32	109.91	106.71	124.51
64	107.03	116.16	130.21

Table 6. Parallel 3 Final Population Convergence Metrics for H20 with NPOP=256 and NGEN=1000

Metric D over CF and N — Metric R over CF and N

CF	N=32	N=64	N=128
4	.112E-2	.111E-2	.261E-2
8	.165E-2	.103E-2	.241E-2
16	.154E-2	.235E-2	.727E-2
32	.182E-2	.257E-2	.949E-2
64	.926E-2	.630E-2	.222E-1

CF	N=32	N=64	N=128
4	.716E+0	.653E+0	.145E+1
8	.922E+0	.830E+0	.121E+1
16	.615E+0	.187E+1	.291E+1
32	.169E+1	.194E+1	.599E+1
64	.327E+1	.283E+1	.865E+1

Table 7 compares E for Parallel 2 and Parallel 3 when addressing the standard cost H20 (H20/std) problem, illustrating the general reduced efficiency obtained when analysis cost is low, and relatively stronger Parallel 3 performance when compared to Parallel 2 in that case. This gain is the result of distribution of Gn as well as the now significant C and Cp over communication domain, characteristics clearly reflected in Tables 8 and 9, comparing O and %M for Parallel 2 and Parallel 3 when addressing the ten-fold cost H20/x10 problem. Here O reduction is greater for larger N and smaller CF, and remains significant for moderate CF with larger N. For example, O reduction is 50 percent when N = 128 and CF = 16. The now significant %M of Parallel 2 is similarly reduced. Unexpected large %M values for smaller N and larger CF are thought to be manifestations of communication contention occurring in the non-dedicated test environment.

Table 7. Parallel 2 versus Parallel 3 Scalability for H20/std with NPOP=256 and NGEN=1000

Parallel 2: E over CF and N — Parallel 3: E over CF and N

CF	N=32	N=64	N=128
4	70.60	44.22	16.46
8	80.64	61.02	27.97
16	88.63	75.61	43.56
32	90.79	85.56	60.20
64	95.22	92.27	74.65

CF	N=32	N=64	N=128
4	81.35	62.23	31.52
8	87.93	73.48	40.88
16	88.59	83.08	52.88
32	92.59	89.79	68.28
64	93.43	94.78	80.28

Table 8. Parallel 2 versus Parallel 3 BCB Specific Cost in Seconds for H20/x10 with NPOP=256 and NGEN=1000

Parallel 2: O over CF and N

CF	N=16	N=32	N=64	N=128
4	14.30	9.25	6.78	5.54
8	12.90	7.22	4.64	3.46
16	11.30	6.44	3.58	2.40
32	11.80	5.58	3.02	1.86
64	10.60	5.45	2.85	1.61

Parallel 3: O over CF and N

CF	N=16	N=32	N=64	N=128
4	12.80	6.84	4.01	2.40
8	11.60	6.08	3.29	1.88
16	11.20	5.58	2.95	1.61
32	10.80	5.38	2.75	1.48
64	10.40	5.54	2.71	1.41

Table 9. Parallel 2 versus Parallel 3 Percent Communications Cost for H20/x10 with NPOP=256 and NGEN=1000

Parallel 2: %M over CF and N

CF	N=16	N=32	N=64	N=128
4	1.40	3.22	8.92	34.54
8	0.76	2.05	5.36	21.62
16	0.55	0.64	2.72	12.17
32	0.17	3.60	1.94	6.79
64	6.42	3.90	1.08	3.78

Parallel 3: %M over CF and N

CF	N=16	N=32	N=64	N=128
4	0.94	3.46	4.80	16.74
8	0.36	2.48	3.64	11.01
16	0.26	2.10	3.57	6.88
32	1.46	1.45	1.60	3.99
64	5.16	2.31	2.02	2.21

Table10 demonstrates Parallel 3 performance gains when addressing higher cost analyses. Problem H20/99p, whose analysis cost on a single processor represents 99 percent of the total execution cost, scales to 128 processors with near 99 percent efficiency when employing near minimal acceptable communication frequency. Parallel 2 demonstrates similar relative efficiency gains when addressing higher cost analyses.

Table 10. Parallel 3 Scalability Comparison for H20/x10 and H20/99p with NPOP=256 and NGEN=1000

H20/x10: E over CF and N

CF	N=32	N=64	N=128
4	94.50	93.44	78.31
8	95.73	95.67	85.62
16	97.07	95.45	91.14
32	98.70	98.45	94.99
64	94.96	95.84	97.67

H20/99p: E over CF and N

CF	N=32	N=64	N=128
4	98.68	98.53	92.50
8	95.98	99.51	95.82
16	99.24	100.08	97.61
32	98.24	100.08	99.02
64	99.62	98.53	99.70

Tables 11 and 12 compare Parallel 2 and Parallel 3 performance when addressing the larger 80-member hub frame problem. Table 11 indicates Parallel 3 to be somewhat less tolerant of infrequent communication than Parallel 2. Comparison with Table 5 demonstrates a reduced degradation in robustness for this larger problem, a characteristic attributed to the need for a larger generation limit, NGEN, a position substantiated by the need for an NGEN well beyond 1000 for the successful application of the approximate Kuhn-Tucker termination criterion described later. Table 12 presents scalability results for Parallel 2 and Parallel 3 when applied to H80/x10. Comparison with Table 10 illustrates the extended scalability of Parallel 3 when applied to a larger problem. This characteristic was observed for standard cost problems and with Parallel 2 as well. This improvement appears to be driven by the dominance of computation growth over communication growth, manifested in significantly reduced communications overhead (%M metric).

Table 11. Parallel 2 versus Parallel 3 Robustness for H80 with NPOP=256 and NGEN=2000

Parallel 2: U over CF and N

Parallel 3: U over CF and N

Benchmark (N=1) Volume (U=100) = 93900

CF	N=32	N=64	N=128
4	100.24	100.24	100.35
8	99.35	103.43	97.05
16	104.32	98.84	107.48
32	99.11	98.52	106.63
64	96.47	105.70	117.81

CF	N=32	N=64	N=128
4	98.79	104.64	102.53
8	97.80	101.91	106.84
16	98.63	97.04	106.63
32	96.45	103.38	117.41
64	104.03	107.46	120.97

Table 12. Parallel 2 versus Parallel 3 Scalability for H80/x10 with NPOP=256 and NGEN=2000

Parallel 2: E over CF and N

Parallel 3: E over CF and N

CF	N=32	N=64	N=128
4	94.72	89.49	78.20
8	97.16	94.26	87.03
16	98.63	97.06	93.05
32	99.28	97.56	96.25
64	99.00	99.18	97.23

CF	N=32	N=64	N=128
4	96.50	95.61	91.21
8	98.51	97.29	94.55
16	98.91	97.99	96.61
32	98.70	98.60	97.72
64	98.16	99.28	97.39

Table 13 compares the scalability of Parallel 2 and Parallel 3 when applied to a ten-fold simulated cost version of a 180-member hub structure (H180/x10) representing 1080 design variable. While population (NPOP) remains at 256 per generation, a smaller generation limit (NGEN) of 513 is used, one just sufficient for an unbiased evaluation of Parallel 3 communication costs over the addressed ranges of N and CF. The superior efficiency of Parallel 3, particularly for larger N, is clearly shown. A comparison of these results with the corresponding efficiency data for H80/x10 in Table 12 indicates this superiority of Parallel 3 to be maintained with further increase in problem size. For N = 128, both parallel versions demonstrate increased scalability with increased problem size. Additional trends of interest include decreased E with increased N under Parallel 2 for CF = 4, but a reversal of that trend when CF increases to 64. Also, under Parallel 3, for N = 128 E increases above the E values in Parallel 2 when CF increases.

Table 13. Parallel 2 versus Parallel 3 Scalability for H180/x10 with NPOP=256 and NGEN=513

Parallel 2: E over CF and N

Parallel 3: E over CF and N

CF	N=32	N=64	N=128
4	93.54	88.51	77.76
8	96.26	93.45	87.47
16	97.58	92.13	92.90
32	97.90	83.83	92.19
64	94.45	96.22	97.31

CF	N=32	N=64	N=128
4	96.87	94.75	92.19
8	98.65	96.05	95.50
16	99.00	96.36	97.60
32	97.31	96.19	98.76
64	99.50	96.64	99.27

Parallel 3 also served as the technique for comparing two alternative termination criteria, providing the potential for near optimal solution detection in a minimal number of generations. The first, OD, is based on satisfying an upper bound on the standard deviation of an (average value) normalized objective population, while the second, KT, is based on a geometric approximation of Kuhn-Tucker criterion satisfaction which does not require the computation of any Lagrange multipliers. KT equates Kuhn-Tucker satisfaction with the ability to express the negative objective gradient as a linear combination of one or more positively scaled gradients of a set of critical constraint when no constraint are violated. Set members include all functional and side constraint whose magnitudes are less than a designated near-zero upper bound. Preliminary KT validation was facilitated by generally successful application to inequality constrained problems of the Hock and Schittkowski test set (ref. 5) along solution paths of the optimization code DONLP2 by Spellucci (ref. 6).

Given a design space of dimension NDV, the KT algorithm addresses a sequence of NDV*(NDV-1)/2 two-dimensional design subspace spanned by all unique pairs of the NDV coordinate axes. A geometric analysis within each subspace attempts to identify Kuhn-Tucker criterion failure by determining an inability to express the projection of the negative objective gradient, PNGF, as either a linear combination of any two positively scaled projection of critical constraint gradient, PGG, or close alignment with a single such projection. The linear combination determination is based on identifying a PGG bounded critical sector of less than PI radians, which includes PNGF. Computations involve normalized projection components only, with no transcendental evaluations required. While any two-dimensional Kuhn-Tucker criterion failure immediately identifies KT termination test failure, test success requires completion of all two-dimensional analyses with criterion success.

Table 14 compares the effectiveness of OD with KT on the H20 problem. Included are results for DK, a hybrid approach reducing overhead by delaying use of KT until a relaxed OD (D1) criterion is satisfied. For these tests the generation limit, NGEN, was increased to 4000. D1 and D2 represent OD criteria using upper bound, D, values of .0001 and .00005 respectively. The critical constraint bound used with KT and DK was 0.033. As with OD, smaller values of this bound delay KT criteria satisfaction. Larger values of this bound, on the other hand, potentially admit more constraints into the critical set, leading to earlier satisfaction. For these tests, KT termination satisfaction is more sensitive to N than that of OD. The OD and KT control bounds represented in Table 14 are near optimal for the problem addressed. These results demonstrate success with both criteria, and identify the superiority of OD, and significant cost overhead of KT, found to be dominated by the cost of calculating the derivatives. DK efficiency is better than KT, but still not competitive with OD. Projected cost reduction through parallelization of the finite difference portions of hub frame semi-analytic derivative computations is insufficient to render KT or DK cost competitive, particularly for larger N.

Table 14. Termination Criteria Comparison Using Parallel 3 on H20/std with CF=16, NPOP=256, NGEN=4000

TTe' over Criteria Type and N

TYPE	N=16	N=32	N=64
None	100.00	100.00	100.00
D1	37.85	45.07	63.04
D2	46.74	58.19	73.60
KT	42.12	87.04	136.38
DK	39.38	76.62	113.41

E' over Criteria Type and N

TYPE	N=16	N=32	N=64
None	100.00	100.00	100.00
D1	96.87	99.02	98.59
D2	98.97	98.68	97.50
KT	89.83	90.94	80.58
DK	95.43	95.93	89.98

U over Criteria Type and N

TYPE	N=16	N=32	N=64
None	101.51	105.06	100.98
D1	102.31	106.12	101.51
D2	101.86	105.68	101.24
KT	102.48	105.24	100.98
DK	102.31	105.24	100.98

G over Criteria Type and N

TYPE	N=16	N=32	N=64
None	4000	4000	4000
D1	1473	1793	2497
D2	1857	2305	2881
KT	1377	2881	4000
DK	1473	2881	4000

Table 15 presents averaged robustness and scalability data for 15 replications of H20/std solution with Parallel 3, using a random distribution of initial sampling seed. Comparison with corresponding Table 5 and Table 7 data, obtained with the single seed value on which all above results are based, substantiate these results. The improved benchmark (BM) volume of Table 15 reflects the use of the previously mentioned more effective sampling technique. Variation of U over range of initial seed was typically ten percent of BM. Table 15 E values are significantly improved over corresponding Parallel 3 E values of Table 7, and can be traced to increased communication efficiency with longer (replicated solution) communication sequences, suggesting an improved amortization of communication startup costs.

Table 15. Parallel 3 Replication Averaged Performance for H20/std With NPOP=256, NGEN=1000, and Improved Sampling

Robustness: U over CF and N
BM Volume (N=100) = 11900

CF	N=32	N=64	N=128
4	99.83	102.78	104.80
8	103.03	103.87	106.40
16	105.39	104.88	112.12
32	105.47	109.34	119.02
64	109.26	114.48	127.44

Scalability: E over CF and N

CF	N=32	N=64	N=128
4	83.54	70.32	46.15
8	91.35	81.97	59.31
16	94.70	89.18	71.85
32	97.78	93.50	80.96
64	98.83	96.20	87.60

Population size, NPOP, is identified in the Implementation section as a critical factor for our parallel implementations of BCB. These require a minimum processor local population of two, and larger (e.g. four) to maintain robustness. While the modification of Parallel 3 identified there promises to extend parallel computation with robustness to N = NPOP processors (with some additional localized communication), additional scalability based on a coarse-grained distribution of problem analysis requires a larger population.

The effectiveness of increasing formal scalability with a larger population depends on converging to a robust solution in a corresponding reduced number of generations. Table 16 compares the volume of H20 solution volume, V, computed by Parallel 2 with a fixed number of total problem analyses, NGEN*NPOP = 256,000, over a varying number, NGEN, of generations. In this case, the results obtained with N = 1, and N = 8 with CF = 1, 8, and 64 all indicate optimal V to occur with NGEN between 2000 and 4000, suggesting an optimal N between 64 and 128. The significantly larger increases in V obtained with larger than optimal NPOP, compared with those obtained with smaller than optimal NPOP, indicate optimal NPOP over-estimation to be more detrimental to BCB performance than under-estimation. Additional testing with NPOP values of 256, 512, and 1024 on up to 128 processors substantiated these results, demonstrating effective parallel BCB scalability of a given hub frame problem to be limited by a critical population size. Our experience with hub frame problem of larger size suggests a corresponding increase in critical population sizes.

Table 16. Parallel 2 Computed V Variation with NGEN for a Fixed Number of Total Analyses

NGEN	NPOP	N=1	N=8/CF=1	N=8/CF=8	N=8/CF=64
16000	16	12608	13704	12645	13634
8000	32	13340	13623	13337	13997
4000	64	12608	11425	12651	13239
2000	128	13064	11759	12830	12563
1000	256	12806	14033	13832	13029
500	512	14467	14271	14282	14412
250	1024	16376	16972	16502	16075

In regard to time saved, Ts, it is apparent that in case of an optimization by GA that time depends on the number of individual designs in a population and on the number of generations in the entire GA process. Let us now refer for an example to the case represented by Table 13. Assuming the highest efficiency from that table of nearly 100 percent and 1 minute for analysis of a single design, we can estimate the elapsed time for one generation analysis to be 1 minute instead of 256 minutes (4.3 hours) that would be needed on a single processor. On the other hand, the total elapsed time for the entire GA optimization performed on 256 processors involving 513 consecutive generations would require 513 minutes (8.6 hours), typically an overnight run. However, the same operation on a single processor machine would occupy 131,328 minutes (2189 hours, 91 days), a time prohibitively long.

This example vividly shows that a multiprocessor implementation of a GA algorithm may make a difference between doing it or not even trying in case of a large application. It also illustrates the discrete nature of practical time saving. For example, a reduction in total elapsed time to 12 hours would still allow overnight execution, while a reduction to 18 hours would not, suggesting an increased difference in practical significance between 12 and 18 hour execution times compared with 8.6 and 12 hour times. Similar distinctions can be made with respect to other time scales, such as one versus five-minute turn around for interactive processing. The example suggests also that if the number of generations can be reduced by increase in the population size, that trade-off should be exploited to reduce the total elapsed time, providing a sufficient number of processors are available. Such compression of GA elapsed time is limited, however, by the need to progress through a certain number of generations. That number cannot be reduced to one by expansion of population size. This sets a limit on the number of processor that can be effectively engaged, as demonstrated by Table 16. Finally, one should point out that detailed examination of GA optimization history reveals that the number of generation can be reduced by terminating the process as soon as an individual design sufficiently close to a constrained minimum is detected. This is demonstrated in Table 14 and points to the need for a reliable criterion to terminate a GA process.

Summary and Concluding Remarks

Numerical experiments were conducted with an Evolutionary Algorithm (a category of Genetic Algorithms) for optimization to verify expectations that that algorithm is a natural for implementation on a machine with many processors. The Evolutionary Algorithm was based on a Gaussian probability distribution in its reproductive mechanism and was introduced in ref. 2. The test was a hub structure of up to 180 members reported in ref. 4. As many as 128 processors were used simultaneously. Parallel algorithm implementations were successful in closely approximating serial benchmark solution quality.

Three parallel implementations of an existing Bell-Curve Based Evolutionary Optimization (BCB) code were evaluated for robustness and scalability against hub frame problems of increasing size and computational cost. The first version, employing serial child generation and a single communication domain, is limited in scalability by the dominant BCB-specific computation, parent selection and child generation. Relative to parent selection, child generation is much more costly and amenable to parallelization, making it the natural target for extended parallelization.

The second version, replacing serial child generation with a parallelized equivalent within each population partition, results in a dramatically improved scalability whose principal limit is communication overhead. This version tolerates well global communication of reduced frequency, with communication intervals up to 32 generations maintaining average solution degradation within five percent and maximum degradation within ten percent of the serial benchmark for the 20-member hub frame problem.

The third and final parallel version provides scalability beyond the above versions by replacing global (single domain) communication, with communications within mutually exclusive sub-domain, of varied granularity. In effect, it treats sub-domain specific processor subsets as single virtual processors of larger size (and larger associated BCB population) within which local population evolution under decreased isolation is maintained. For a given communication frequency, the reduction of the communication cost relative to the other two techniques increases with the number of processors used. Parallel 3 robustness compares well with that of Parallel 2, when measured in terms of solution degradation. One may conclude that multi-domain communication strategies reduce communication overhead with limited impact on robustness.

The results show that, indeed, a multiprocessor execution may radically reduce the elapsed time for the entire GA optimization process. That reduction may enable large GA optimization applications that could not have even been attempted on single processor machines. The need for GA to progress through a certain number of generations, however, limits the extent to which elapsed time may be reduced by distributing larger populations over more processors.

On the other hand, the non-distributable part of the algorithm, and the processor-to-processor communication generated as a result of parallel execution were shown to be factors that severely limit the number of processors that may be used efficiently. In GA, these limits tend to diminish with the increase of the cost of the design analysis but they do not vanish. Therefore, it was determined that to mitigate the detrimental effect of the processor-to-processor communication, it is imperative to devise techniques that strictly control and reduce the extent of the processor-to-processor data transmissions. The Parallel 3 technique, which employs a multiple domain communication strategy to so limit data transmission cost, demonstrates efficiencies exceeding 99 percent on 128 processors in some cases. As might be expected, GA optimization robustness, measured in terms of solution quality for fixed number of design analyses over a fixed number of generations, suffers when reduced communication frequency and increased population partitioning curtail communication. In the test case, processor local populations of at least four and communication frequencies of at least every 32 generations were needed to limit average solution degradation to five percent.

There is some potential for further reduction of the elapsed time to be realized by terminating the process as soon as there is one design generated sufficiently close to a constrained minimum. Two termination criteria tested are population objective distribution (OD) and approximate Kuhn-Tucker (KT) satisfaction based. OD criterion satisfaction occurs when the normalized standard deviation of population objective falls below a threshold. KT criterion satisfaction is based on a geometric interpretation of the Kuhn-Tucker criterion that avoids computation of Lagrange multipliers. That interpretation depends on ascertaining both an absence of violated constraints and the ability to express the negative objective gradient as a linear combination of admissible critical constraint gradients, where admissibility is determined by an upper bound on constraint magnitude. Sensitivity control is accomplished by OD threshold or KT bound parameter adjustment. Although these criteria demonstrate similar effectiveness, the derivative requirement for KT significantly increases overhead, reducing overall efficiency. KT also displays more sensitivity to increased N. OD is recommended over KT as a GA termination criterion.

Finally, it should be noted that effective and efficient multiprocessor computing requires the method developer to learn about the hardware/software architecture to be used to a much greater extent than it was necessary for a conventional single processor implementation.

References

1. Sobieszczanski-Sobieski, J., Laba, K., and Kincaid, R. K., "Bell- Curve Based Evolutionary Optimization Algorithm", in Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis MO., September 2-4, 1998. AIAA Paper 98-4971, pp. 2083-2096.

2. Kincaid, R. K., Weber, M., and Sobieszczanski-Sobieski, J., "Performance of a Bell-Curve Based Evolutionary Optimization Algorithm" AIAA Structures Dynamics and Materials Conference, Atlanta, April 2000.

3. Amdahl, G.M. "Validity of the Single-processor Approach to Achieving Large Scale Computing Capabilities", AFIPS Conference Proceedings vol. 3 (Atlantic City, N.J., Apr. 18-20). AFIPS Press, Reston, Va., 1967, pp.483-485.

4. Balling, R. J., Sobieszczanski-Sobieski, J. 1994, "An Algorithm for Solving the System-Level Problem in Multilevel Optimization", ICASE Report No. 94-96 and NASA Contractor Report 19501 December 1994.

5. Hock, W., and Schittkowski, K.; "Test Examples for Nonlinear Programming Codes", Lecture Notes in Economics and Mathematical Systems 187, Springer, Berlib-Heidelberg-New York 1981.

6. P. Spellucci; Resources downloaded from URL http://plato.la.asu.edu/donlp2.html, Test environment file "testenviron.tar.gz", Code and documentation file "donlp2.tat.gz".

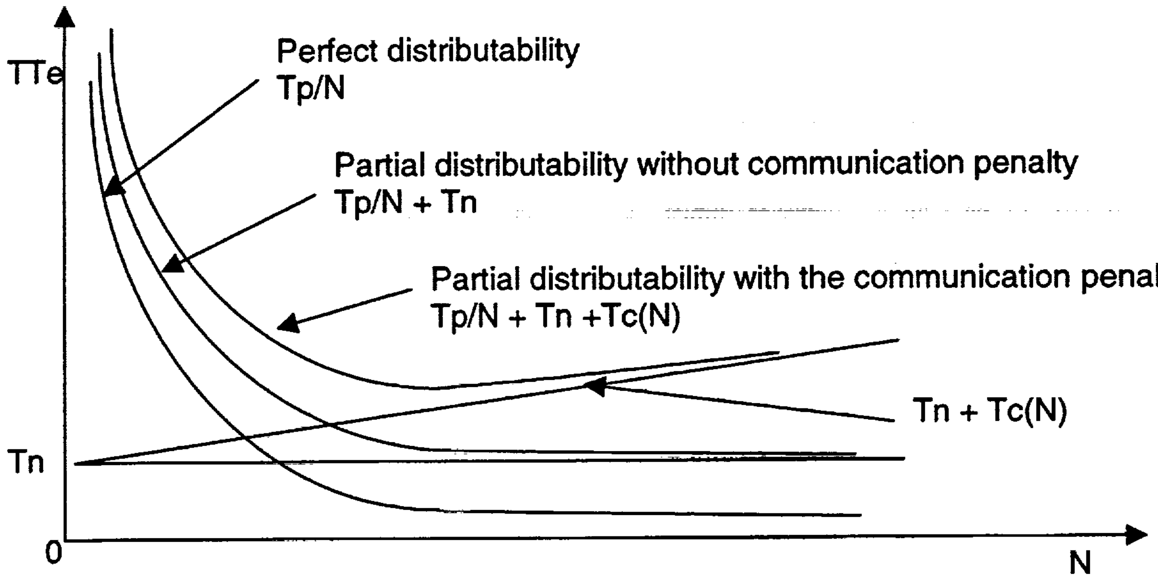

Figure 1. Elapsed time of a computation on multiprocessor machine vs. number of processors

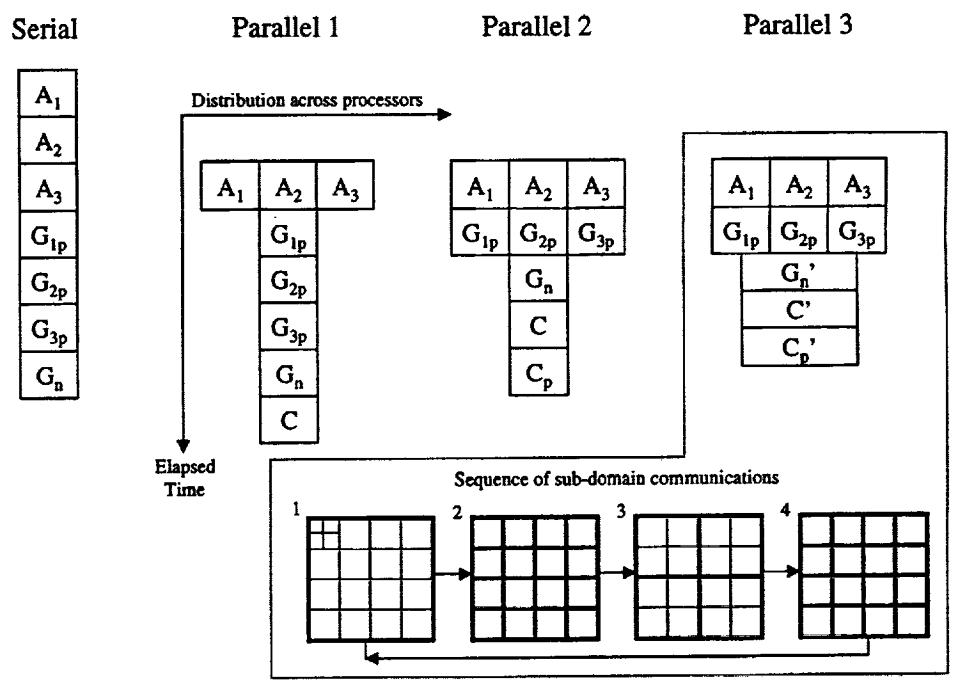

Figure 2. Serial and Three Parallel Techniques for Execution