Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence (FOCI 2007)

Hybrid Optimisation Method Using PGA and SQP Algorithm

B. T. Skinner, H. T. Nguyen, D. K. Liu

Abstract— **This paper investigates the hybridisation of two very different optimisation methods, namely the Parallel Genetic Algorithm (PGA) and Sequential Quadratic Programming (SQP) Algorithm. The different characteristics of genetic-based and traditional quadratic programming-based methods are discussed and to what extent the hybrid method can benefit the solving of optimisation problems with nonlinear complex objective and constraint functions. Experiments show the hybrid method effectively combines the robust and global search property of Parallel Genetic Algorithms with the high convergence velocity of the Sequential Quadratic Programming Algorithm, thereby reducing computation time, maintaining robustness and increasing solution quality.**

Index Terms— **Parallel Genetic Algorithm, Sequential Quadratic Programming Algorithm, Hybrid Methods, Global Optimisation, Evolutionary Algorithms and Constraint Functions.**

I. Introduction

PARALLEL Genetic algorithms (PGAs) have become an increasingly popular method for solving global optimisation problems [1-3]. They are very effective techniques for searching complex problem spaces for an optimum. Parallel and Sequential Genetic Algorithms [4] are stochastic algorithms that require minimal information from the problem domain to guide the search process. These methods perform a search by evolving a random population of tentative solutions towards the optimum solution. This is achieved through the iterative application of simple and stochastic operators inspired from those of genetics (e.g., selection, crossover and mutation). Genetic algorithms provide significant advantages over traditional search methods including implicit population-wide search (not greedy), ability to control exploration (population diversity) and exploitation (convergence) and the principle of building-block combination (reuse of search information) [5, 6].

Traditional optimisation methods, such as the Sequential Quadratic Programming (SQP) Algorithm [7] and Quasi-Newton Method exploit all local information in an efficient way, provided the objective function to be minimised is "well-conditioned" in the neighbourhood of the optimum [8]. Otherwise, these methods tend to converge to local optima when used to solve complex optimisation problems containing numerous local optima. Conversely, genetic algorithms experience slow convergence before providing an accurate solution because the minimal use of *a priori* domain knowledge and not exploiting local information.

This paper presents a hybrid search algorithm that couples the benefits offered by both classes of search technique. The proposed hybrid search algorithm combines the reliable exploratory property of a parallel genetic algorithm with the efficient exploitative property offered by the reduced feasible sequential quadratic programming algorithm (RFSQP) [9, 10], in order to reduce the overall computation time. To illustrate the theoretical aspects and facilitate an empirical comparison of the hybrid search algorithm, a suite of multimodal non-linear functions have been selected. We compare the convergence velocity, reliability, solution quality (accuracy) and computation time of the standalone parallel genetic algorithm to the hybrid search algorithm.

II. The Hybrid Search Algorithm

The hybrid search algorithm aims to combine the parallel genetic algorithm (PGA) and reduced feasible sequential quadratic programming algorithm (RFSQP) in order to blend their advantages and minimise their disadvantages. The hybrid search algorithm presented in this paper belongs to the family of heuristic search algorithms described by memetic algorithms[11], also called genetic local search (GA+LS) algorithms.

Specifically, the hybrid search algorithm allows a global search to be performed using a cascaded architecture with the PGA in the primary stage followed by the RFSQP algorithm in the secondary stage as illustrated in Figure 1. The cascaded architecture enables the hybrid search algorithm to initially explore the entire search space for promising regions and then exploit these sub-spaces while satisfying and required constraint functions.

A. Exploration Stage – PGA

Large, multi-dimensional, multi-modal global search problems are typically searched using precedence and balance between exploration and exploitation. In the primary stage of the hybrid search algorithm it is important to explore the search space for promising sub-spaces, also called

Manuscript submitted January 14, 2007. This work was supported by the Mechatronics and Intelligent Systems (MIS) Group, University of Technology, Sydney, Australia. The authors are with the MIS Group, University of Technology, Sydney, 2000, Australia. (ph)+612 95142451, (fax) +612 95142868, (e-mail: brad.skinner@uts.edu.au).

neighbourhoods, that may contain the global optimum (minimum or maximum), without prematurely converging to a local optimum. The robust, explorative and non-greedy properties of the parallel genetic algorithm are important during the initial stage of the hybrid search. Once the parallel genetic algorithm has narrowed the search region to a promising neighbourhood of the search space that contains the global optimum, evolution is halted and the solution vector, $\mathbf{X}_i$ of the overall fittest individual provides input to the secondary stage of the hybrid search algorithm.

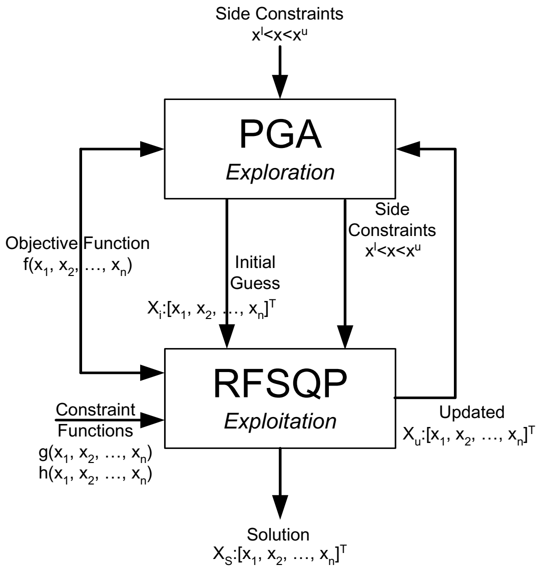

Figure 1: The hybridised search algorithm architecture. The PGA is coupled with the RFSQP algorithm using feedback. The feedback mechanism allows the PGA to received 'only improving' solution vectors from the RFSQP.

The exploration stage of the hybrid search algorithm is handled exclusively by the PGA. To perform fitness evaluations of individuals in a population, the PGA works directly with objective function. The search space or fitness landscape is bounded by initial side constraints. No other *a priori* knowledge is provided to the parallel genetic algorithm.

In general, the parallel genetic algorithm begins the search with a population of randomly generated candidate solutions from the search space. This initial population is evolved iteratively using stochastic genetic operators so most of the population in later generations have reached the neighbourhood containing the global optimum. In this paper, the parallel genetic algorithm is multiple-deme and coarse-grained with loosely-coupled, static subpopulations.

Decoding of chromosome genotype from binary-coded to a real-coded phenotype representation is required for all fitness evaluations. All subpopulations evolve in isolation starting from an initial random population of candidate solutions. All subpopulations maintain two non-overlapping populations, with each new generation replacing the old to help maintain population diversity (generational model). To minimise the likelihood of premature convergence of the PGA in each subpopulation, elitism is not used, prior to the subpopulation being updated from the RFSQP phase as discussed in section II-B.

For all subpopulations, the *tournament selection* method is used to select a fixed number of *tour* individuals chosen randomly from the population. The fittest individual from the *tour* is then selected for mating. This process occurs twice with two, different parents chosen for mating. The tournament selection method allows selective intensity and population diversity to be adjusted through setting of the *tour* size$\left(2 \le tour \le popsize\right)$. The recombination of binary encoded chromosomes is performed using uniform crossover or multi-point crossover, depending on the particular objective function being solved. Both crossover methods use the same crossover probability (P_c=0.95).

Bit-flip mutation is performed with an adaptive probability given by Equation (1).

$$p_m\left(t\right) = \left(\frac{n \cdot l}{20} + \frac{2 \cdot n \cdot l}{T-1} \cdot t\right)^{-1} \tag{1}$$

where, $p_m(t)$ is the temporal mutation probability at generation step t, $t \in \left\{0,1,...,T-1\right\}$, T is the maximum number of generations in a single epoch, n is the problem dimensionality, and l is the gene bit length (chromosome length is nl).

The adaptive bit-flip mutation described in Equation (1) provides a large $p_m(t)$ in the early stage of evolution for aiding the exploration of the search space and a small $p_m(t)$ in the later stage to protect highly fit individuals from random mutations, since they usually have 'more to lose' in variation of the chromosome, thus retaining good solutions [12]. The genes of each child chromosome are decoded into corresponding phenotype values, for direct use with objective function evaluation and determination of fitness. Each chromosome contains a number of genes corresponding to the number of function dimensions in each test function.

Migration between subpopulations is governed by the migration policy. The migration policy specifies the migration frequency, rate and selection/insertion criteria of individuals between subpopulations as illustrated in Figure 2. All subpopulations evolve in isolation until the number of generations is equal to the migration frequency. The migration frequency determines when migrations occur between subpopulations. Individuals are selected based upon their phenotype fitness value. The fittest individuals are selected and migrated to the master process. The master process evaluates all migrants, including its own, to determine the overall fittest migrant. The least-fittest individuals in each subpopulation are replaced by the overall fittest migrant. This fitness based selection and replacement strategy provides a higher selection pressure opposed to a random strategy for the parallel genetic algorithm [13].

The proportion of individuals selected from a subpopulation and sent during a migration is specified by the migration rate. Migrations between subpopulations and the master process are unordered and blocking, which may introduce small wait states in the hybrid algorithm, during periods of migration. The parallel genetic algorithm is halted before reaching the global optimum. The solution (design) vector, $\mathbf{X}_i$, of the overall fittest individual is provided to the secondary stage RFSQP algorithm. The PGA halts computation once the objective function fitness value reaches a predefined and static proportion of the global optimum. This halting value is typically defined as 99.9%, 99.99% or 99.999% of the global optimum value.

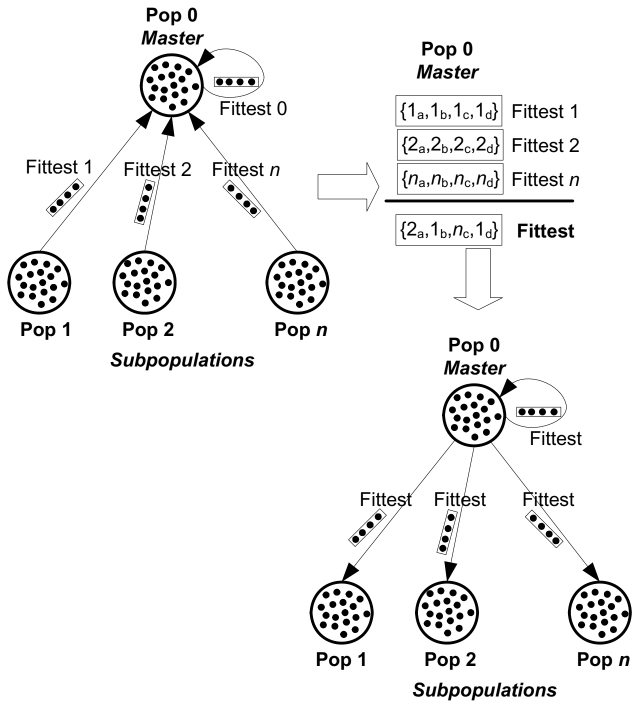

Figure 2: Unordered and blocking communication provides synchronous migration between loosely-coupled subpopulations. The fittest individuals are migrated to a master process for final fitness evaluation and retransmission to all subpopulations, replacing least-fit individuals.

B. Exploitation Stage - RFSQP

The secondary stage of the hybrid search algorithm is handled exclusively by the RFSQP algorithm, which provides its highly exploitative (convergence) properties. In this paper, the RFSQP algorithm takes as input the objective function, 'best' solution vector, $\mathbf{X}_i$ as the initial guess, and any side constraints. Although, the RFSQP algorithm can enforce equality and inequality constraints, this feature is not described in this paper.

The RFSQP algorithm uses local information to guide a search of the neighbourhood supplied by the parallel genetic algorithm in the solution vector $\mathbf{X}_i$. Typically, the RFSQP algorithm has a very high convergence velocity and rapid computational time, but it may become trapped in local optima if the neighbourhood is large.

The RFSQP algorithm expands the objective function quadratically about the current design vector and linearly expands the constraint functions, thus establishing one QP problem, and two linear least squares problems. The quadratic programming and linear least squares subproblems are then solved using QPOPT. For all QPs the active set in the solution at a given iteration is used as the initial guess for the active set in the solution at the next iteration.

The RFSQP algorithm halts once stopping criteria has been fulfilled. If the RFSQP algorithm managed to locate the global optimum the hybrid search algorithm terminates and $\mathbf{X}_s$ represents the final solution vector. If the RFSQP algorithm simply moved closer to the region containing the global optimum, without locating the global optimum, the updated solution vector, $\mathbf{X}_u$ is feedback to the primary stage parallel genetic algorithm where it is evaluated for fitness. If the corresponding phenotype fitness value of the $\mathbf{X}_u$ solution vector is greater than the phenotype fitness value of the $\mathbf{X}_s$ solution vector (i.e. an improvement was made), $\mathbf{X}_u$ is encoded into a chromosome and *injected* into the PGA population. Once the highly fit chromosome of $\mathbf{X}_u$ has been injected into the PGA population, a high degree of elitism is enabled to ensure that it permeates to other PGA subpopulations during subsequent migration periods. For the case where fitness($\mathbf{X}_u$) is less than fitness($\mathbf{X}_s$), the encoded $\mathbf{X}_u$ solution vector is *not* injected into the population and the PGA continues until the next call to the RFSQP algorithm.

The hybrid search algorithm continues to iterate through the loop of parallel genetic algorithm followed by the RFSQP algorithm at fixed stages of the hybrid search with the PGA population moving closer to the global optimum.

III. EXPERIMENT DESCRIPTION

A set of multimodal, non-linear, smooth objective functions are used to illustrate the theoretical aspects and facilitate an empirical comparison between the parallel genetic algorithm and the hybrid search algorithm.

A. Comparison Criteria

The comparison criteria include convergence velocity, solution quality, reliability and computation time when searching for the global optimum. Essentially, these criteria indicate the effectiveness of the search algorithm to balance the exploration and exploitation trade-off.

1) *Convergence Velocity*: Practically, convergence velocity is measured by observing the rate and magnitude of change between terms of a sequence. Here, the phenotype fitness values of the objective function at each generation represents the sequence. Furthermore, it is not usually satisfactory to halt a search algorithm based exclusively on the magnitude of the difference between successive values of the objective function, because the search procedure would stop prematurely for regions of the search space having a "gentle-slope".
2) *Solution Quality*: Solution quality, often called accuracy

is a measure of the distance from the final solution to the global optimum, for searches that do not result in failure. For the PGA and hybrid search algorithm solution quality is given by the fitness value of the fittest individual in the population once the algorithm has terminated.

3) *Reliability*: The reliability of a search method is the ability to locate the global optima and avoid local optima. In this study, an objective function fitness value $\geq 1.0e^{-13}$ at the maximum number of generations is considered a failure for the search method. Each objective function and search method is executed for 50 independent experiments. Accordingly, the reliability of a search method is characterised by the ability to locate solutions where $fitness \leq 1.0e^{-13}$ for 50 independent experiments.
4) *Computation Time*: The computation time for search algorithms is often a measure of the number of objective function evaluations, because different algorithms may be compared regardless of their particular implementation. However, this often disregards communication times for parallel implementations. In this study, the performance is measured by recording the wall-clock time, so all components of the execution time, including communications, are included. The wall-clock time is a fair measure of performance that is frequently used [13].

B. Numerical Test Functions

To facilitate an empirical comparison of the parallel genetic algorithm and the hybrid search algorithm a test environment is provided in the form of a set of idealised objective functions, expressed in closed analytical form. The test environment contains functions used extensively in evolutionary computation literature to benchmark evolutionary algorithms including the Sphere Function (f_{Sph}), Ackley's Path Function (f_{Ack}), Rastrigin's Function (f_{Ras}), Schwefel's Function (f_{Sch}), Michalewicz Function (f_{Mic}) and the Langerman Function (f_{Lan}). Optimisation is characterised by searching for a global minimum of each objective functions given in Appendix A.

C. Parallel Genetic Algorithm Input Parameters

There is a complex, non-linear relationship between the parameters of a parallel genetic algorithm which influences behaviour [6]. Table I lists the input parameters for the parallel genetic algorithm. To encourage a meaningful empirical comparison between the parallel genetic algorithm and hybrid search algorithm, the set of genetic parameters remain constant in both algorithms. These parameters are not applicable to the SQP algorithm.

Computation time for the parallel genetic algorithm and hybrid search algorithm is halted once the fittest individual has a phenotype fitness value $\leq 1.00 \times 10^{-13}$ or all objective function evaluations have been performed.

D. RFSQP Algorithm Input Parameters

Table II lists the input parameters for the RFSQP algorithm. These parameters are not applicable to the parallel genetic algorithm. Additional algorithm parameters specified in the file, *param.h*, remain at their default values in [10], for all experiments.

TABLE I
PARALLEL GENETIC ALGORITHM PARAMETER SET

	f_{Sph}	f_{Ras}	f_{Ack}	f_{Sch}	f_{Mic}	f_{Lan}
Generations	400	400	600	600	600	400
Population Size	200	200	200	400	400	400
Populations	8	8	8	8	8	8
Aggregate Population	1600	1600	1600	3200	3200	3200
Migration Freq	4	4	4	25	10	4
Migration Rate	0.01	0.04	0.01	0.005	0.005	0.005
Migrant Selection	Fittest	Fittest	Fittest	Fittest	Fittest	Fittest
Migrant Reinsertion	Least-fit	Least-fit	Least-fit	Least-fit	Least-fit	Least-fit
Gene bit size			53			52
Dimensions	30	20	30	10	10	5
Chromosome **Length(bits)**	1590	1060	1590	530	530	260
Tour Size	5	3	6	2	2	6
$\boldsymbol{P_m}$			Adaptive (Eq4)			
$\boldsymbol{P_c}$			0.95			
X-over Sites	Uni	3	Uni	8	8	4

TABLE II
REDUCED FEASIBLE
SEQUENTIAL QUADRATIC PROGRAMMING ALGORITHM
PARAMETER SET

RFSQP Parameter	***Value***
Problem Dimensions (nparam)	(Table I)
# Objective Functions (nf)	1
# Equality constraint functions (neq)	0
# Inequality constraint functions (neq)	0
Maximum Iterations (miter)	200
Stopping Criteria (epsneq)– Sum of abs values of non equality constraints	$7.0e^{-10}$
Number of sets of affine SR constraints (nclsr)	0
Number of sets of nonlinear SR constraints (ncnsr)	0
Number of sets SR objectives (nosr)	0
Bound for QP subproblems (bigbnd)	$1.0e^{15}$
Current design vector – initial guess – (sqpx)	bestindividual.x[i]
Lower Bound Side Constraints – (bl)	phenotype_lb
Upper Bound Side Constraints – (bu)	phenotype_ub
Start/Stopping Criteria (eps) - $\boldsymbol{f_{sph}}$	0.990 / $10e^{-7}$
Start/Stopping Criteria (eps) – $\boldsymbol{f_{Ras}}$	0.999 / $10e^{-5}$
Start/Stopping Criteria (eps) – $\boldsymbol{f_{Ack}}$	0.990 / $10e^{-8}$
Start/Stopping Criteria (eps) – $\boldsymbol{f_{Sch}}$	0.990 / $10e^{-8}$
Start/Stopping Criteria (eps) – $\boldsymbol{f_{Mic}}$	0.999 / $10e^{-7}$
Start/Stopping Criteria (eps) – $\boldsymbol{f_{Lan}}$	0.900 / $10e^{-7}$

We found experimentally that limiting the maximum number of RFSQP iterations to 200 gave good results for all objective functions, while minimising the RFSQP computation.

E. Cluster Computing Environment

All experiments were performed on a computing cluster with specifications provided in Table III. The parallel genetic algorithm is coded in C++ and employs the Message Passing Interface Specification ver2.0 for communication between subpopulations running on separate cluster nodes. The RFSQP algorithm was coded in C.

TABLE III
CLUSTER COMPUTING HARDWARE AND SOFTWARE ENVIRONMENT

Computing Environment Component	Description
Number of Node Utilised	8
Processor Type and core Speed	Pentium 4 Hyper threading @ 3.6Ghz (Prescott)
Front-side Bus Bandwidth	800MHz
DRAM capacity and bandwidth	2GB DDR2 @ 533MHz
Network Type and Bandwidth	1000Mbps Ethernet
Network Switching Type	Gigabit Switching Fabric
Network Protocol	TCP/IP V4
OS Kernel Type and Version	Linux (2.4.21-20.EL)
MPI Type and Version	LAM 7.0.6 / MPI 2
Compiler Type and Version	mpiCC and gcc (3.2.3)
Coding Language Standard	ISO C++ (PGA) and C (RFSQP)

IV. RESULTS AND DISCUSSIONS

Experimental results for the PGA and hybrid search algorithm are illustrated in Figure 3 to Figure 8. Each figure is a graph representing *Objective Function Fitness* value versus *Generation* count using a log-linear scale. The horizontal dotted line located at an objective function fitness value of $1.0e^{-13}$ is the stopping criterion and minimum solution needed for a successful search. For all graphs, each curve represents the average of the best result achieved from 50 independent experiments.

During the early stage of a search the convergence velocity is approximately the same for both algorithms in all test functions. This is expected, since the hybrid search algorithm uses an identical PGA in the primary stage for exploration of the search space. In general, the curves begin to deviate once the hybrid algorithm makes the transition from primary stage PGA to secondary stage RFSQP algorithm.

Furthermore, results of the hybrid search algorithm show periods of extremely rapid convergence in the form of *cliffs*. These cliffs initially occur at different stages of the search for each test function and are the result of the secondary stage RFSQP algorithm. The difference in objective function fitness value between the solution vector provided by the primary stage PGA (top of cliff) and the resulting solution vector found by the RFSQP algorithm (base of cliff), is proportional to the effectiveness of the RFSQP algorithm in moving closer to the global optimum.

The RFSQP algorithm is limited to 200 iterations of the objective function ensuring computation is complete within a single generation of the parallel genetic algorithm. The graphs provide a valid representation, because total computation time of the secondary stage RFSQP algorithm is less than or equal to the total time required for the computation of a single PGA generation.

Results for the Sphere Function show a single cliff extending beyond the stopping criterion and exceeding the PGA solution quality as shown in Figure 3. Because the Sphere Function is unimodal and smooth, the RFSQP algorithm converges directly to the global optimum once provided with the initial guess by the primary stage PGA. No more iterations of the hybrid search algorithm are required.

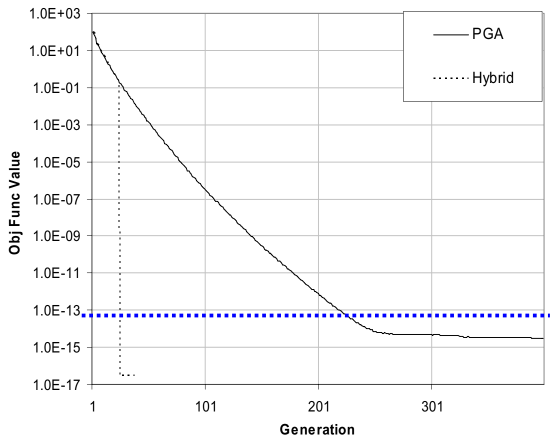

Figure 3: PGA and Hybrid Algorithm using the Sphere Function

Figure 4 illustrates results for the multimodal Rastrigin Function. Essentially, a single cliff is followed by a declining rate of convergence, as additional iterations of the PGA are required to reach the stopping criterion, due to the cosine modulation producing many local minima. For the same reason, further iterations of the PGA and RFSQP algorithm are also needed to achieve similar solution quality as the PGA.

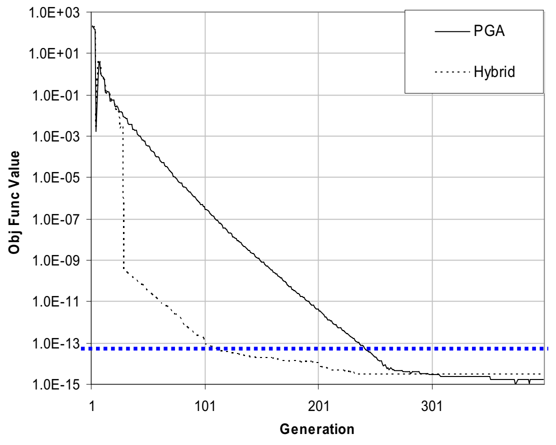

Figure 4: PGA and Hybrid Algorithm using the Rastrigin Function

The Ackley Path function in Figure 5, has three distinct cliffs. There exists one large cliff in the early stage of the search as the secondary stage RFSQP algorithm away moves off the large plateau and falls towards the global optimum. Two smaller cliffs occur at generation 200 and 300 as the RFSQP algorithm makes small improvements due to many local optima. The PGA and RFSQP algorithm are called three times to achieve the stopping criterion and to exceed the PGA solution quality. The complex multimodal landscape of the Ackley Path Function prevents the RFSQP algorithm directly locating the global optimum, does provide the PGA with higher fitness solution early in the search.

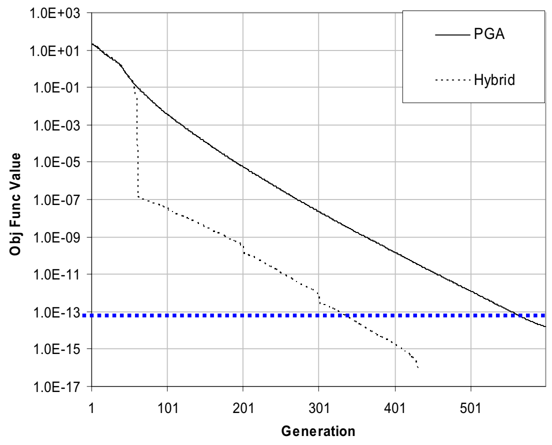

Figure 5: PGA and Hybrid Algorithm using the Ackley Path Function

Results for the continuous Schwefel Function show a single cliff extending very close to the stopping criterion in Figure 6. This result suggests that the initial guess provided to the RFSQP algorithm is definitely located in the neighbourhood of the global optimum. Although, further iterations of the PGA and RFSQP algorithm are required to reached the stopping criterion and eventually exceed the solution quality of the PGA. The geometrically distant global optimum that is not oriented along the axes gives a slow convergence velocity in the later stage of the search, with no significant improvements being made by the RFSQP algorithm.

Figure 7 illustrates results for the Michalewicz Function, which is continuous and multimodal. The fitness landscape of the Michalewicz Function contains steep gullies or edges interconnected with plateaus containing no gradient information causing transition from PGA to RFSQP algorithm to occur at different generations during the search for each of the 50 experiments. This can be seen as the curve for the hybrid search algorithm *wanders* after deviating from the PGA curve. In other words, the primary stage PGA falls from the plateau into the gulley containing the global optimum at different stages of the search.

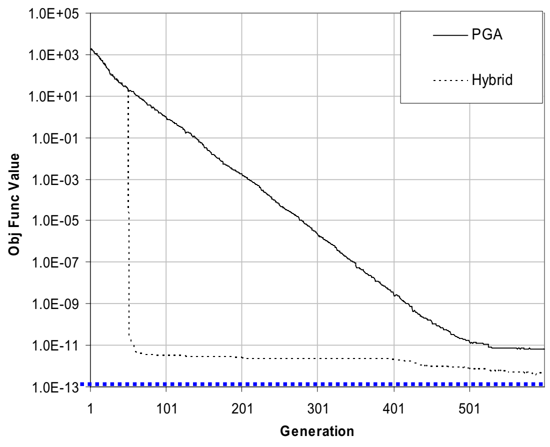

Figure 6: PGA and Hybrid Algorithm using the Schwefel Function

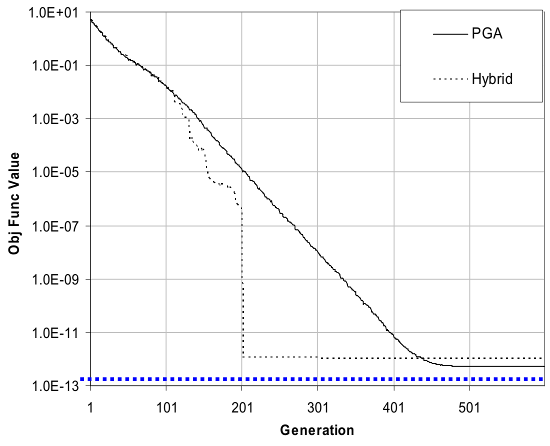

Figure 7: PGA and Hybrid Algorithm using the Michalewicz Function

Results for the continuous and highly multimodal Langerman Function show a single cliff extending beyond the stopping criterion and exceeding the PGA solution quality as shown in Figure 8. The RFSQP algorithm converges directly to the global optimum once provided with the initial guess by the primary stage PGA. No more iterations of the hybrid search algorithm are required. The *wandering* phenomenon also occurs for the Langerman Function, since many local minima are unevenly distributed in the fitness landscape, which causes the PGA to enter the promising neighbourhood at different generations for the 50 experiments.

For each of the 50 independent experiments the hybrid search algorithm was able to locate the global optimum for all test functions, and maintain the robust search properties of the standalone PGA.

TABLE IV
PERFORMANCE RESULTS FOR THE PGA AND HYBRID SEARCH ALGORITHM

		f_{Sph}		f_{Ras}		f_{Ack}	
		PGA	**Hybrid**	**PGA**	**Hybrid**	**PGA**	**Hybrid**
Computation	*Mean*	4.780	0.593	2.833	1.068	12.962	6.303
Time (Seconds)	*Stdev*	0.812	0.151	0.201	0.185	0.304	1.024
Reliability	*Mean*	183	20	208	75	508	244
(Generations)	*Stdev*	30	6	13	14	10	39

		f_{Sch}		f_{Mic}		f_{Lan}	
		PGA	**Hybrid**	**PGA**	**Hybrid**	**PGA**	**Hybrid**
Computation	*Mean*	7.755	5.295	6.196	7.718	1.682	0.459
Time (Seconds)	*Stdev*	0.358	2.011	0.381	2.363	0.657	0.103
Reliability	*Mean*	538	341	431	120	93	18
(Generations)	*Stdev*	24	152	25	26	19	5

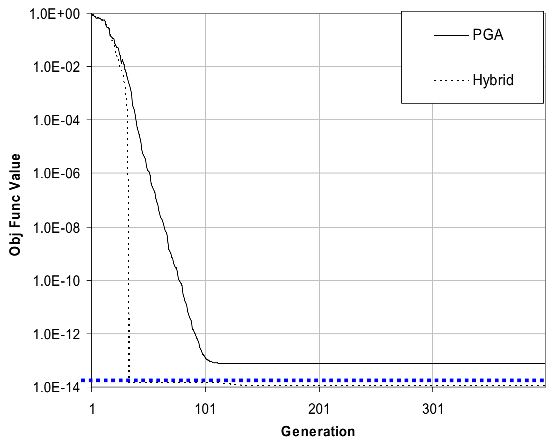

Figure 8: PGA and Hybrid Algorithm using the Langerman Function

Table IV lists the performance results of the parallel genetic algorithm and hybrid search algorithm for each test function. The termination criterion, for all experiments, is the ability of the algorithm to reach an objective function fitness value of $1.0e^{-13}$, which determines the computation time and reliability metrics listed in Table IV.

Computation time is reduced for each test function, except the Michalewicz function due to an outlier in the experimental results, which delayed convergence of the hybrid search algorithm and thus resulted in a large standard deviation of 2.363. Both algorithms reached the termination value for all 50 independent experiments, but the hybrid search algorithm achieved this using far less generations for each test function.

V. CONCLUSIONS AND FURTHER INVESTIGATIONS

This paper presented a performance evaluation of a hybrid search algorithm which combined two very different optimisation techniques. Simulation experiments show the hybrid search method effectively combines the robust search property of the parallel genetic algorithm with the high convergence velocity of the RFSQP Algorithm.

The *injection* feedback method, which couples the output of the secondary stage RFSQP algorithm to the input of the primary stage parallel genetic algorithm, enabled the hybrid search algorithm to significantly reduce computation time, increase solution quality, and maintain robustness.

Currently, the hybrid search algorithm performs the transition from PGA to RFSQP algorithm using predefined and static values. Future research would aim to follow the progress of the primary stage PGA by tracking the population diversity, convergence velocity and/or computation time. Taken together this dynamic method would remove the need for artificial values and provide a more 'natural' approach in switching between the exploration and exploitation phases.

To provide a comparison between the hybrid search method and the RFSQP algorithm, future investigation will examine a stratified multi-start RFSQP algorithm with and without a limitation on the number of iterations from the same initial solution for each numerical test function presented in this paper.

In addition, replacement of the existing binary-coded chromosomes with real-valued chromosomes will remove the PGA phenotype-to-genotype (real-to-binary) decoding step, which currently uses between 33% and 45% of PGA computation time. Also, constrained optimisation problems could be fully defined through enforcement of equality and inequality constraint functions in the secondary stage RFSQP algorithm.

ACKNOWLEDGMENT

I wish to thank Dr Matthew Gaston and Mr Peter Brady for establishing, maintaining and supporting the UTS Engineering Linux Computing Cluster Environment.

APPENDIX A

TABLE V
NUMERICAL TEST FUNCTIONS DEFINITIONS

Function Definition
(1) $f_{Sph}\left(x\right)=\sum_{i=1}^{n}\left(x_i^2\right)$
(2) $f_{Ras}\left(x\right)=n\cdot A+\sum_{i=1}^{n}\left(x_i^2-A\cos\left(2\pi x_i\right)\right)$
(3) $f_{Ack}\left(x\right)=20+e-20\cdot\exp\left(-0.2\sqrt{\frac{1}{n}\cdot\sum_{i=1}^{n}\left(x_i\right)^2}\right)-\exp\left(\frac{1}{n}\cdot\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)$
(4) $f_{Sch}\left(x\right)=\sum_{i=1}^{n}\left(-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$
(5) $f_{Mic}\left(x\right)=-\sum_{i=1}^{n}\sin\left(x_i\right)\cdot\left(\sin\left[\left(i+1\right)\cdot x_i^{\frac{2}{\pi}}\right]\right)^{2m}$
(6) $f_{Lan}\left(x\right)=-\sum_{i=1}^{m}\mathrm{c_i}\cdot\left(e^{-\frac{1}{\pi}\sum_{j=1}^{n}\left[x_j-A_{ij}\right]^2}\cdot\cos\left(\pi\sum_{j=1}^{n}\left[x_j-A_{ij}\right]^2\right)\right)$

TABLE VI
NUMERICAL TEST FUNCTIONS RANGE AND VALUE

	Function Range	Function Value
(1)	$-5.10\le x_i\le 5.10,$ $n=30,i=1:n$	$f_{Sph}\left(\tilde{x}\right)=0.0,$ $x_i=0.0$
(2)	$-5.10\le x_i\le 5.10,$ $A=10,n=20,i=1:n$	$f_{Ras}\left(\tilde{x}\right)=0.0,$ $x_i=0.0$
(3)	$-32.8\le x_i\le 32.8,$ $e=\exp\left(1\right),n=30,$ $i=1:n$	$f_{Ack}\left(\tilde{x}\right)=0.0,$ $x_i=0.0$
(4)	$-500\le x_i\le 500,$ $n=10,i=1:n$	$f_{Sch}\left(\tilde{x}\right)=-n\cdot 418.98288,$ $x_i=420.96874636$
(5)	$0.0\le x_i\le\pi,n=10,$ $m=10,i=1:n$	$f_{Mic}\left(\tilde{x}\right)=$ -9.660151715, $x_i=?$
(6)	$0.0\le x_i\le 10,n=5,$ $m=30,i=1:m,j=1:n,$	$f_{Lan}\left(\tilde{x}\right)=$ -1.499943824, $x_i=?$

REFERENCES

[1] E. Cantú-Paz, "A Survey of Parallel Genetic Algorithms," Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 S. Mathews Avenue Urbana, IL 61801, Report 97003, May 1997.

[2] E. Alba and M. Tomassini, "Parallelism and Evolutionary Algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 6, pp. 443 - 462, 2002.

[3] Z. Konfrst, "Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspectives," presented at Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, Santa Fe, New Mexico, USA, 2004.

[4] J. H. Holland, *Adaptation in natural and artificial systems*, 1st MIT Press ed. ed. Cambridge, Mass: MIT Press, 1992.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, vol. 1, 16 ed: Addison Wesley Longman, Inc, 1997.

[6] M. Mitchell, *An Introduction to Genetic Algorithms*, vol. 1, 2 ed: Bradford, 1996.

[7] P. Venkataraman, *Applied Optimisation with Matlab Programming*, vol. 1, 1 ed. Canada and New York: John Wiley and Sons, 2002.

[8] J.-M. Renders and S. Flasse, "Hybrid methods using genetic algorithms for global optimization," *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, vol. 26, pp. 243-258, 1996.

[9] C. Lawrence and A. Tits, "A Computationally Efficient Feasible Sequential Quadratic Programming Algorithm," *SIAM J. Optimization*, vol. 11, pp. 1092-1118, 2001.

[10] C. Lawrence and A. Tits, "RFSQP User's Guide," AEM Design May 9 2002.

[11] P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms," California Instiitute of Technology, Pasadena, Report No 826, 1989.

[12] M. Glickman and K. Sycara, "Reasons for Premature Convergence of Self-Adapting Mutation Rates," presented at Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, La Jolla, CA USA, 2000.

[13] E. Cantú-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, 1 ed: Kluwer Academic Publishers, 2000.