6. The new Price's algorithm for a DAP

The algorithm described in Section 4 was modified to remove the three main sequential sections of code. The main details of the new algorithm are shown in Figure 2 but briefly they are as listed below:

(i) After rejecting the largest \( n \) points on each processor we 'compact down' the matrices \( M_1 \) and \( M_2 \) (holding the coordinates and function values of points respectively) by removing the rejected values, this means we no longer need to take care to avoid them.

(ii) Instead of choosing the overall 4096 maximum values, we now select 4096, but the maximum from each processor, which can be done quite simply in parallel.

(iii) Instead of choosing \( n \) points at random, one from each processor, we now choose \( n \) matrices at random. This means that it is now no longer necessary to record the positions of the randomly selected points for later reference, since they have all come from one of \( n \) matrices.

(iv) Since we are now choosing whole matrices at random rather than individual points, then the replacement of old points by new ones is a lot simpler, i.e. we can generate a logical mask, containing TRUE values, only at positions where the new point is contained within the domain \( S \) AND the function value \( f_{P_1} \) is less than \( f_{M_1} \). This mask then being used in a series of simple assignments.

(v) When we compacted down the two matrices there was space left over, \( n \) planes in \( M_1 \), \( n^2 \) in \( M_2 \), this space is now used for holding the centroids, new trial points, maximum values and the function values of new trial points.

(vi) The random number initialization routine and generator were also changed to faster but adequate routines.

set of \( 2n \cdot 4096 \) and together with the minimum point \( f_{L_1} \), determine the centroid followed by the new trial point matrix \( P \).

i.e.

Suppose we have chosen \( n \) points \( R_1, R_2, \ldots, R_{n+1} \) at random and \( R_1 = L \) then

\[
\bar{R} = \frac{1}{n} \sum_{i=1}^{n} R_i
\]

and

\[
P = 2\bar{R} - R_{n+1}
\]

If the new points generated are within the domain \( S \) then we evaluate \( F(P) \) otherwise we must repeat by choosing more points at random, for those which are not within \( S \), we also repeat from this stage if \( F(P) \) is worse than the existing point.

At this stage (after one or possibly more iterations) we have a full set of 4096 points which are contained within the domain \( S \) and whose function values are less than the corresponding maximum function values.

It is these improved values that are used in the replacement process and the test for termination is carried out.

The termination criterion examines the 3 smallest values of the function values \( L_1, L_2, L_3 \) such that \( L_1 < L_2 < L_3 \) then if \( |L_1 - L_2| < |L_2 - L_3| \) and \( |L_2 - L_3| < \varepsilon \) (\( \varepsilon \) = some tolerance) we stop, otherwise repeat by selecting the 4096 largest values etc.

The algorithm also contains as a safety measure an upper bound on the number of iterations (in this case 40 - purely an arbitrary figure).
In order to generate new test points, we choose 906 generator points at

In the previous section, we discussed how to choose 906 generator points. Now, let's look at the next steps. We will use these points to generate new test points.

1. Choose new test points
2. Calculate the distance between new test points and existing generator points.
3. Select new test points based on the distance.
4. Repeat for all generator points.

The complete process involves selecting new test points based on the distance from existing generator points. This process is repeated for all generator points.

Let's consider the next step. We will use the selected new test points to generate new test points. This process is repeated for all generator points.

In summary, the process involves selecting new test points and generating new test points based on the distance from existing generator points. This process is repeated for all generator points.

2. System for the Implementation

---

The system for the implementation involves selecting new test points and generating new test points based on the distance from existing generator points. This process is repeated for all generator points. The system is designed to efficiently generate new test points in a systematic manner.

---

Note: The diagram and text are provided for reference. The actual implementation may vary based on specific requirements.
7. Final results

The results of this new version of the algorithm are also displayed in Table 1 (sequential vs. 'new' parallel). This time a positive improvement can be seen compared with the sequential algorithm and an even greater improvement on the 'old' parallel algorithm.

A direct comparison of the times for the sequential and the new parallel yields the improvement factor which can be seen to range from 3.2 up to 68. This considerable range can be explained by the fact that the lower improvement factors came from functions which had just one global minimum, whereas the larger factors came from the functions possessing multiple global minima. The best example is probably the SHUBERT function which has 18 global minima; the 'new' parallel algorithm picked up 11 of these minima in one run, whereas the sequential algorithm required 10 runs to pick up these same 11 minima.

It can also be seen that the number of parallel function evaluations needed has been drastically reduced in most cases to single figures but at the most to just 14.

Table 2 contains the numeric results of the programs and Table 3 the actual values (obtained from [5] and elsewhere). It can be seen that there is a reasonable comparison in all of the test cases except for SHUBERT's function where the algorithm only managed to locate the approximate position of the global minima, this was then allowed to run for 40 iterations and also 8 different random sequences but still failed to improve upon the quoted figure.

Taking the percentage differences of the values we find that the results for the sequential algorithm were within an average of 0.6% of the actual values and for the new parallel algorithm within an average of 1.54% but if we include the results from the SHUBERT function then this figure worsens to 4.57%.

1. The Problem

The problem of global optimisation is as follows:

$\min F(x) \quad x \in S \subset \mathbb{R}^n$

where $S = \{ x : a_i \leq x_i \leq b_i \}$

Assuming that the function $F(x)$ is non-convex and has more than one local minimum, then the problem is to isolate the point $x^*$ with the least function value (which is by definition, the global minimum).

An international study comparing the relative efficiency of a number of programs for the solution of the global optimisation problem is reported in Dixon & Szego [5]. That study indicated that most of the efficient codes are based on probabilistic principles and that the heuristic CRS algorithm proposed by Price [6] was one of the most successful. He has recently proposed a simplified version [7] and this was chosen for implementation on the parallel system.

The probabilistic method of solution relies on the following assumption:

if we have a finite number of points chosen at random and uniformly distributed in the region $S$, then if $A$ is a subset of $S$ with a measure $\mu$ such that

$\frac{\mu(A)}{\mu(S)} \geq \alpha > 0$

and $p(A,N)$ being the probability that at least one point of a sequence of $N$ random points lies in $A$,

then $\lim_{N \to \infty} p(A,N) = 1$

The heuristic method then aims at improving the best points located in the random search.
The partial evaluation of the function with complex parameters can be performed by taking into account the initial implementation of the code section (i.e., the code with the initial implementation) in the same manner as in the actual implementation. This approach allows for the allocation of a number of features to the code section, which can be used to optimize the overall performance.

The approach is to use an efficient technique for partial evaluation and an efficient algorithm for partial evaluation. This would involve the use of a combination of techniques to optimize the implementation of the code section, which can be used to improve the overall performance of the system.

In the implementation, the partial evaluation is only applicable for small values of the input data. However, this approach could be extended to larger values of the input data through the use of efficient algorithms.

The implementation of the partial evaluation for the function with a complex parameter can be performed by taking into account the partial evaluation of the function with the initial implementation. This approach allows for the allocation of a number of features to the code section, which can be used to optimize the overall performance.

With the use of the partial evaluation, the implementation of the function with a complex parameter can be performed by taking into account the partial evaluation of the function with the initial implementation. This approach allows for the allocation of a number of features to the code section, which can be used to optimize the overall performance.
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>SEQUENTIAL</th>
<th>NEW PARALLEL</th>
<th>OLD PARALLEL</th>
<th>IMPROVEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>$T_N^N$ eval.</td>
<td>Time (s)</td>
<td>$T_N^N$ eval.</td>
</tr>
<tr>
<td>SIX HUMP</td>
<td>0.36</td>
<td>288</td>
<td>0.093</td>
<td>7</td>
</tr>
<tr>
<td>CAMEL BACK</td>
<td>0.81</td>
<td>798</td>
<td></td>
<td>Note (1)</td>
</tr>
<tr>
<td>(2 global minima)</td>
<td>1.17</td>
<td>1086</td>
<td>Note (1)</td>
<td></td>
</tr>
<tr>
<td>BROWN</td>
<td>0.47</td>
<td>360</td>
<td>0.118</td>
<td>8</td>
</tr>
<tr>
<td>(3 global minima)</td>
<td>1.37</td>
<td>654</td>
<td></td>
<td>Note (1)</td>
</tr>
<tr>
<td></td>
<td>3.17</td>
<td>1024</td>
<td></td>
<td>Note (2)</td>
</tr>
<tr>
<td>GOLSTEIN &amp; PRICH (1</td>
<td>0.45</td>
<td>401</td>
<td>0.13</td>
<td>8</td>
</tr>
<tr>
<td>global minimum)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NARTMAN'S</td>
<td>0.88</td>
<td>503</td>
<td>0.276</td>
<td>11</td>
</tr>
<tr>
<td>N = 3 (1 global minimum)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N = 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHERKER</td>
<td>3.89</td>
<td>2732</td>
<td>0.453</td>
<td>14</td>
</tr>
<tr>
<td>(1 global minimum)</td>
<td>Note (5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 5</td>
<td>3.82</td>
<td>2423</td>
<td>0.55</td>
<td>14</td>
</tr>
<tr>
<td>m = 7</td>
<td>Note (5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 10</td>
<td>4.31</td>
<td>2567</td>
<td>0.693</td>
<td>14</td>
</tr>
<tr>
<td>SHERBER 2-D (10 global</td>
<td>average</td>
<td>average</td>
<td>0.233</td>
<td>8</td>
</tr>
<tr>
<td>minima)</td>
<td>1.56/min</td>
<td>995</td>
<td>Note (4)</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES**

1. Multiple global minima were identified in one run.
2. Two runs were needed to locate 5 global minima.
3. Improvement of new parallel as compared with the sequential.
4. The parallel algorithm picked up 11 minima in one run hence the same 11 minima from the sequential version were used for the comparison (these 11 being picked up in 10 runs).
5. For this particular function the approximate location only was located for the global minimum.

**TABLE 1**


<table>
<thead>
<tr>
<th>Task</th>
<th>Operation</th>
<th>Parameters</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Addition</td>
<td>A + B</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>Subtraction</td>
<td>A - B</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>Multiplication</td>
<td>A * B</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>Division</td>
<td>A / B</td>
<td>C</td>
</tr>
</tbody>
</table>

### Example Calculations

- **Addition**:
  - 5 + 3 = 8
  - 10 + 5 = 15

- **Subtraction**:
  - 10 - 5 = 5
  - 20 - 10 = 10

- **Multiplication**:
  - 5 * 3 = 15
  - 10 * 5 = 50

- **Division**:
  - 10 / 2 = 5
  - 20 / 5 = 4
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>ACTUAL VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEMP CAMEL BACK</td>
<td>f = 1.0316285 @ 0.08983, -0.7126 and -0.08983, 0.7126</td>
</tr>
<tr>
<td>BRANIN</td>
<td>f = 0.397887 @ 3.14159, 2.275 and 9.2476, 2.475 and -3.14159, 12.275</td>
</tr>
<tr>
<td>GOLDSTEIN &amp; PRICE</td>
<td>f = 3.0 @ 0.0, -1.0</td>
</tr>
<tr>
<td>HARTMANS N = 3</td>
<td>f = -3.68278 @ 0.11484, 0.585515, 0.852951</td>
</tr>
<tr>
<td>SHEEKL N = 4 m = 3</td>
<td>f = -10.532 @ 4.0005, 4.00004, 4.000001</td>
</tr>
<tr>
<td>m = 7</td>
<td>f = -10.4029 @ 4.00052, 4.00078, 3.99943, 3.99957</td>
</tr>
<tr>
<td>m = 10</td>
<td>f = -10.5564 @ 4.00066, 4.00054, 3.99955, 3.99948</td>
</tr>
<tr>
<td>SHUBERT 2-D</td>
<td>f = -186.73091 @ -7.0835, -7.70831 -1.42513, -0.800032 4.85805, -7.0835 -0.800032, 4.85805 -7.70831, -7.0835 4.85805, -0.800032 -1.42513, 5.48286 -7.70831, -0.800032 -7.70831, 5.48286 -0.800032, -7.70831 5.48286, 4.85805</td>
</tr>
</tbody>
</table>

Now a standard tool for solving constrained nonlinear programming problems, but they were not treated during the Summer School in Urbino. The main topics of the new ASI are not design and implementation of software, but the test and application of optimization algorithms, and should therefore be considered as the continuation of the work, the Committee on Algorithms started in 1977.

In the terminology used by NATO, the director of the Advanced Study Institute is Klaus Schittkowski (University of Stuttgart, Germany, F.R.). Co-directors are Karla Hoffman (National Bureau of Standards, Washington D.C., USA) and Jan Telgen (Rabobank Nederland, Zeist, Netherlands). Limited funds are available for participants from NATO-countries to cover a part of their travel and accommodation expenses. Since the number of participants is limited to approximately 70 persons, participation is possible only by personal invitation. For more information and an application form, write to

Klaus Schittkowski
Institut für Informatik
Universität Stuttgart
Azenbergstraße 12
D-7000 Stuttgart 1
Germany, F.R.
To give an example, sequential quadratic programming methods are often employed in the context of optimization problems where the number of nonlinear constraints increases with the size of the problem. Sequential quadratic programming methods are particularly useful in problems with a large number of variables and constraints, as they allow for efficient computation and provide a good balance between accuracy and computational cost. These methods are often used in engineering design, economics, and operations research, among other fields.

In addition to sequential quadratic programming, there are other optimization techniques available, such as interior-point methods, which are particularly effective for large-scale problems. These methods are known for their ability to handle a wide range of problem types, including those with non-smooth or non-convex objectives. Another important class of optimization methods is the class of metaheuristic algorithms, which are particularly useful for solving complex, large-scale problems that cannot be solved by traditional optimization methods.

Research in optimization continues to evolve, with new methods and algorithms being developed to address the challenges of modern data-driven applications. These developments are essential for advancing the field of optimization and ensuring that it remains a powerful tool for solving real-world problems.
mixed integer programming in mathematical programming systems

the special issue has arrived. anyone interested in a copy of this report should send a self addressed, stamped envelope (weight: 4 oz) to:

Dr. Karla Hoffman
Center for Applied Mathematics
National Bureau of Standards
Washington, D.C. 20234

and will receive this exciting publication post haste!

special issue
mixed integer programming in
mathematical programming systems
Richard H.F. Jackson and Richard P. O'Neill, editors

Contents:
Introduction to the Special Issue - O'Neill & Jackson 1
MIP Tactics - Greenberg 3
Mixed Integer Programming with Honeywell's MPS - Despain 12
Mixed Integer Programming in the APEX-III System - Krabek & Sjoquist 17
Large Scale Mixed Integer Programming Experiments with Sperry Univac's Series 1100 FMPS Product - McCall 35
Mixed Integer Programming on Burroughs Computer - Carstens 71
Bloodhound - The Mixed Integer Optimizer of MPSIII - Kreegan & Welch 78
Some features of integer programming in SCICONIC - Beale 82
Panel Discussion chaired by Jackson & O'Neill 91

a joint publication of:
the computer science technical section of ORSA
the committee on algorithms of the mathematical programming society

nato advanced study institute on
computational mathematical programming
Klaus Schittkowski

the committee on algorithms of the mathematical programming society announces a summer school on computational mathematical programming to be held in Bad Windsheim, Germany F.R., from July 23 to August 2, 1984, under the sponsorship of NATO. the advanced study institute intends to bring together optimization specialists developing algorithms or software, and scientists and engineers who use these tools in their modelling efforts. the ASI consists of tutorials emphasizing new mathematical programming algorithms, software products, computational experiments, numerical test results, and practical optimization models. The following list summarizes the topics and the invited lecturers:

Integer programming (M. Beale, SCICON, Milton Keynes, United Kingdom)
Model building in linear and large integer programming (H.P. Williams, University of Edinburgh, United Kingdom)
Networks (D. Klingman, University of Texas, Austin, USA)
Nonlinear programming (R. Fletcher*), University of Dundee, United Kingdom)
Model building and practical implementation aspects in nonlinear programming (P.E. Gill, Stanford University, USA)
Large linear systems (J. Stoer, University of Würzburg, Germany, F.R.)
Large scale nonlinear programming (P. Toint*), University of Namur, Belgium)
Geometric programming (M. Rijckaert, University of Leuven, Belgium)
Nondifferentiable optimization (J. Zowe, University of Bayreuth, Germany, F.R.)
A short communication section consisting of brief and timely announcements of new results, applications, codes or experiments. These submissions benefit from accelerated refereeing. Papers omitting proofs and details are encouraged if accompanied by supporting material establishing the validity of the assertions made.

THE MATHEMATICAL PROGRAMMING SOCIETY

ENROLLMENT

I hereby enroll as a member of the Society for the calendar year 1983.

PLEASE PRINT Name

Mailing address

My subscription to Mathematical Programming is for my personal use and not for the benefit of any library or other institution.

Signed

The dues for 1983 are:

35 Dollars (U.S.A.)

20 Pounds (U.K.)

72 Francs (Switzerland)

230 Francs (France)

85 Marks (Fed. Rep. Germany)

95 Guilders (Netherlands)

Student Application: Does are one-half the above rates. Have a faculty member verify your student status below and send application with dues to Professor John Mulvey, School of Engineering and Applied Science, Princeton University, Princeton, N.J. 08544.

Faculty verifying status

Institution

Please read this application with your dues to:

The Mathematical Programming Society

The International Statistical Institute

2270 AZ Voorburg, Netherlands

Robert R. Meyer

Jan Telgen

EDITORIAL COLUMN

This is the first COAL Newsletter published under the co-editorship of Jan Telgen and Rob Meyer. Final responsibility for production and publication of the Newsletter still resides with Jan Telgen, but both co-editors will solicit and accept papers for publication. Therefore authors may contact either of the co-editors regarding potential contributions for the Newsletter.

On our part we will try to improve upon the readability of the Newsletter in two ways. First, the Newsletter will be produced from camera-ready material that is as uniform in quality and style as possible within our budget. Second, we will do some screening on the papers to be published. Both these points were put into practice starting with this issue.

A discussion is in progress on the position and function of the COAL Newsletter as indicated in our previous issue. In particular, it may not be possible to continue free distribution of the Newsletter to non-members of MPS. The next issue of the Newsletter will contain the viewpoints of the Committee on this matter and a request for reactions from the readership. The final decision on these issues is planned for the next COAL-business meeting to take place at the NATO ASI in Bad Windsheim (see pages 3-5).
ISSUES WILL BE ADDRESSED IN THE NEWSLETTER.

The committee on algorithms is invited to computational

CORAL ORGANIZATIONS

use and documentation.

To meet certain standards of portability, testing, ease of

Addison-Wesley.

mathematics and programming, there are three

The committee on algorithms is interested in computational

EDITORS OF THE NEWSLETTER

2. Committee Chairmen, M.S.

3. Mathematics, D.C.


8. Mathematics, D.C.


12. Mathematics, D.C.


14. Mathematics, D.C.

15. Computer Science.

16. Mathematics, D.C.


18. Mathematics, D.C.


20. Mathematics, D.C.


22. Mathematics, D.C.


24. Mathematics, D.C.
Mathematical Programming Society
Committee on Algorithms Newsletter

No. 9
September 1983

JAN TELGEN BOB MEYER
EDITORS

Contents:
Editorial Column .................................................. 1
Message from the chairman ........................................ 2
NATO Advanced Study Institute on
   Computational Mathematical Programming .................. 3
   Klaus Schittkowski
Generating Test Problems Quasi-Randomly .................... 6
   Bennett L. Fox
The Implementation of a Parallel Version of
   Price's (CRS) Algorithm on an ICL DAP .................... 8
   Paul G. Ducksworth
Mixed Integer Programming in Mathematical Programming
   Systems .................................................... 22
SIAM Conference on Numerical Optimization .................. 23
MPS Membership Form .......................................... 24