2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Parallelization for Space Trajectory Optimization

Martin Schlueter
Institute of Space and Astronautical Science,
Japan Aerospace Exploration Agency,
Sagamihara, Kanagawa 252-5210, Japan
Email: martin@flab.isas.jaxa.jp

Masaharu Munetomo
Information Initiative Center
Hokkaido University Sapporo
Sapporo 060-0811, Japan
Email: munetomo@iic.hokudai.ac.jp

***Abstract*—The impact of parallelization on the optimization process of space mission trajectories is investigated in this contribution. As space mission trajectory reference model, the well known *Cassini1* benchmark, published by the European Space Agency (ESA), is considered and solved here with the MIDACO optimization software. It can be shown that significant speed ups can be gained by applying parallelization.**

Keywords: Space Trajectory, Cassini Mission, MIDACO, Parallelization, openMP

I. Introduction

Interplanetary space mission trajectory optimization is known to be a challenging task. Due to the highly non-linear and non-convex nature of those problems, evolutionary and hybrid algorithms are widely in use for this kind of application (see for example [5], [1], [3] or [15]). In order to investigate and compare the performance of such algorithms for space trajectory design, the European Space Agency (ESA) publishes a database of Global Trajectory Optimization (GTOP) problems (see [4]) and encourages researchers from space engineering and operational research to submit perpetuate best known solutions to these benchmarks. This benchmark set is known as the GTOP database and represents trajectory models of real-world interplanetary space missions. Table I lists the major benchmark problems from the GTOP database together with the number of decision variables, the number of submissions of perpetuate best known solutions and the time between the first and last of such solution submission. The difficulty of these problems is reflected in the time span between the first and last solution submission reported in Table I, which takes between several months and even years.

TABLE I: GTOP database benchmark problems

Benchmark	Variables	Number of submissions	Time between first and last submission
Cassini1	6	3	6 Month
GTOC1*	8	2	13 Month
Messenger*	26	7	52 Month
Cassini2*	22	7	14 Month
Rosetta	22	7	6 Month
Sagas	12	1	-

*Best known solution found by MIDACO

The MIDACO software [14] was developed as a general purpose optimization solver in collaboration with ESA and EADS/Astrium Ltd and has been extensively tested on space applications. MIDACO currently represents the state-of-the-art for interplanetary trajectory optimization (see for example [13]) and holds three unbroken record solutions on the GTOP benchmarks listed in Table I, including the best known solution to the most difficult problem: Messenger. Readers who are interested in the algorithm of MIDACO are referred to [9], where the details of the Ant Colony Optimization (ACO) algorithm of MIDACO can be found. Readers with a particular interest in the constraint handling technique of MIDACO are referred to [10]. Note that the performance strength of MIDACO does not only rely on its fundamental ACO algorithm, but also on its sophisticated software implementation which took over 7 years of development for the current version.

In this contribution the potential benefit of parallelization for space mission planning is investigated. In order to have a rigorous numerical evaluation (with 400 test runs) and taking into account the computational effort to solve the GTOP benchmarks, the focus here is only on the smallest GTOP benchmark problem: Cassini1. The Cassini1 benchmark is based on the Cassini mission launched by NASA in 1997 and which is a famous reference among space trajectories. This mission represents an interplanetary flight from Earth to Saturn using several gravitational swing-by manoeuvres at Venus, Earth and Jupiter. Figure 1 displays the mission trajectory and highlights its major events.

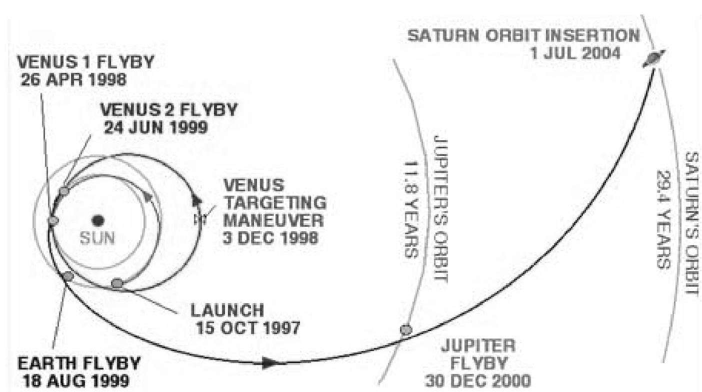

Fig. 1. Interplanetary trajectory of NASA's Cassini Mission (1997)

This paper is structured as follows: In Section II, the parallelization approach by MIDACO is explained and briefly discussed. In Section III, the numerical results by MIDACO for solving the Cassini1 benchmark are presented in regard to the impact of different parallelization factors. The paper finishes with some conclusions and an outlook on possible future research. An Appendix lists detailed information for the numerical results presented in Section III.

II. The parallelization approach by MIDACO

A key feature of the MIDACO optimization software is its calling by *reverse communication*[1] [7]. This means, that the call to the objective function and constraints happens outside the MIDACO source code. This concept does not only guarantee a numerically stable gateway to other languages, but also enables the software to be enriched by a valuable option of distributed computing. Within one reverse communication step MIDACO does accept and returns an arbitrary large number of **P** iterates at once. Hence, those **P** iterates can be evaluated in parallel, outside and independently from the MIDACO source code. This idea of passing a block of **P** iterates at once within one reverse communication step to the optimization algorithm is taken from the code NLPQLP by Schittkowski [8].

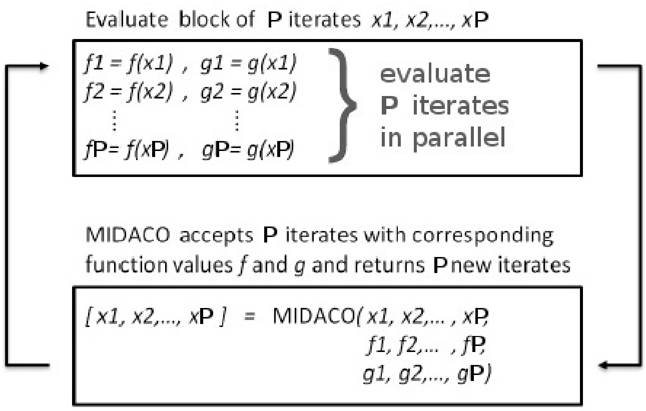

Fig. 2. MIDACO's reverse communication loop over a block of **P** iterates

Figure 2 illustrates the essential reverse communication loop over the function evaluation calls and the MIDACO code. Within one loop, a block of **P** iterates (also called "individuals" in evolutionary algorithms) is evaluated regarding their objective function $f(x)$ and constraints $g(x)$. Then those iterates are passed together with their corresponding objective and constraint values to the MIDACO code. Within MIDACO those iterates are then processed and MIDACO calculates and returns a new block of **P** iterates that needs to be evaluated again. This concept allows an independent and user controlled distributed computing of the objective and constraint function evaluation. In other words, the parallelization option is valid for any language on any CPU architecture without the necessity of adapting the MIDACO source code itself. This includes in particular multi-core PC's, PC-Clusters and GPGPU (General Purpose Graphical Processing Unit) based computation. As the parallelization factor **P** can be arbitrary large, MIDACO is also suitable for massive parallelization.

As this parallelization approach aims on distributing the function evaluation calls, it is intended for problems where the function evaluation are numerically expensive to compute, which is often the case for complex real-world applications. In contrast to this, MIDACO's parallelization approach is not promising for applications, which are very cheap to compute, as the numerical overhead due to the distributed computing exceeds the benefit of parallelized function calls.

For further details on the parallelization approach by MIDACO, please consult [11] or [12]

[1]"I'm not entirely unconvinced that it's not totally evil" Victor Eijkhout on *reverse communication* (Computer Programming Forum, 2002).

III. Numerical Results

This section displays the numerical results obtained by MIDACO on the Cassini1 benchmark taken from the GTOP database [4]. The focus of this section is the influence of the parallelization factor **P** (see Section II) on the MIDACO performance to solve this benchmark. The Cassini1 benchmark problem considers 6 continuous decision variables and 4 non-linear constraints. The best known solution to Cassini1 corresponds to an objective function value of about $f(x) = 4.9307$.

Table III displays the average results of 400 individual test runs on Cassini1 in regard to different parallelization factors **P** of 1,2,4 and 8. Note that **P** = 1 represents the serial mode for MIDACO and acts here as the reference. Because the numerical results in Table II are based on a lot of test runs, a moderate precision of 1% was considered for the success criteria on the objective function value. The lower bounds for all variables were considered as starting point for every test run. All MIDACO parameters were set on default, thus the results represent the **out-of-the-box** performance of MIDACO. All test have been conducted on the same PC with Intel i7 4702MQ CPU with 2.2 GHz clock rate and 8 GB RAM. All source codes were written in C/C++ and compiled with gcc using openMP for parallelization. Note that the considered CPU is a quad-core processor supporting hyper-threading. Therefore speed ups for a parallelization factor of until **P** = 8 are possible under this setup.

The abbreviations for Table II are as follows:

P	Parallelization factor for MIDACO (see Section II)
$f(x)$	Average objective function value for solution x
$Blocks$	Average number of processed blocks (see Section II)
$Eval$	Average number of function evaluation in total
$Time$	Average total CPU-time in Seconds

TABLE II: Average Results for different **P**

P	$f(x)$	$Blocks$	$Eval$	$Time$
1	4.977279	3150654	3150654	99
2	4.977542	1683175	3366349	57
4	4.977675	991475	3965898	38
8	4.977332	458177	3665418	23

The individual details on the results displayed in Table II can be found in the Appendix. The convergence curves of the 100 individual test runs corresponding to average results displayed in Table II are shown in Figure 3, Figure 4, Figure 5 and Figure 6. The average $Time$ is indicated in those figures as vertical red line.

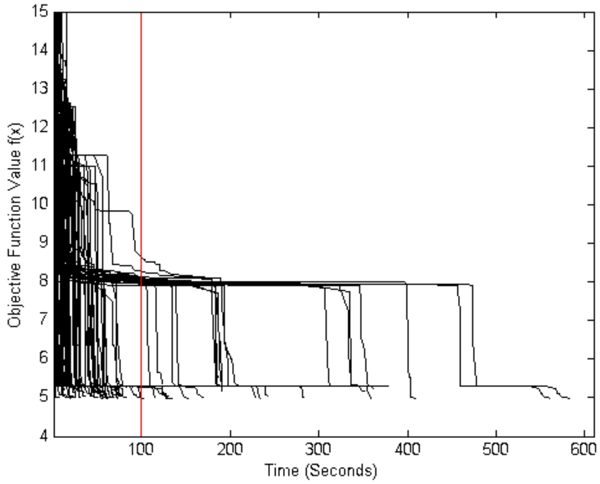

Fig. 3. Convergence curves of 100 test runs with factor **P** = 1

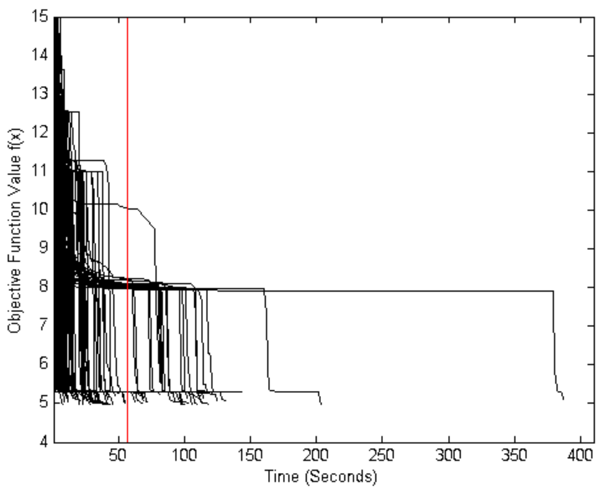

Fig. 4. Convergence curves of 100 test runs with factor **P** = 2

From Figures 3 to 6 it can be observed that only few test runs have exceptional long cpu-runtime behaviour and heavily influence the average results. In all cases, but particular in case of **P**=8, it can be seen that majority of test runs converges to the best known solution well under the average cpu $Time$ indicated as red line.

As mentioned in Section II, the efficiency of the parallelization strategy by MIDACO depends on the cpu-time expense of the problem functions (objective and constraints). While for cheap cpu-time problems MIDACO's parallelization strategy is not promising and might even slow down the overall performance due to an increased overhead by the parallelization effort, significant speed ups for cpu-time expensive applications can be expected.

The cpu-time cost of Cassini1 on the considered CPU takes about 0.00003[1] Seconds, which is fairly cheap but still consid-

[1]Average Time (99) divided by average evaluation (3150654) is about 0.00003

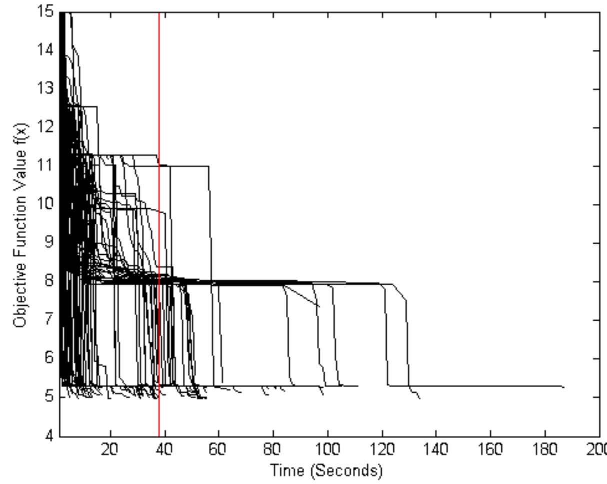

Fig. 5. Convergence curves of 100 test runs with factor **P** = 4

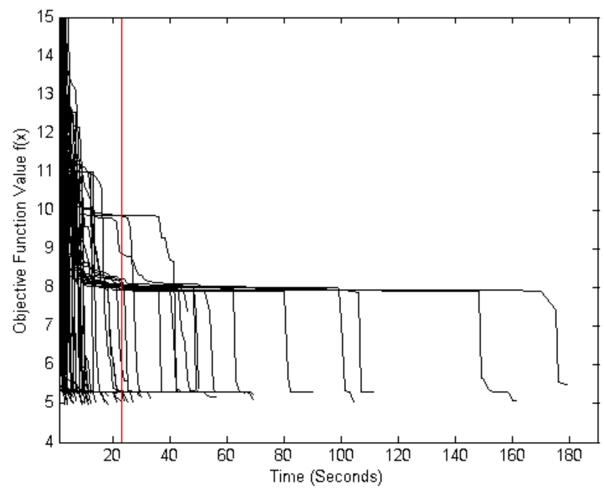

Fig. 6. Convergence curves of 100 test runs with factor **P** = 8

erable more expensive than trivial optimization benchmarks[2]. However, with such a cheap evaluation cost of Cassini1 the effectiveness of MIDACO's parallelization strategy is weakened by the additional overhead of the parallelization effort. Table III compares the potential[3] speed up with the actual[4] measured speed up for the here considered parallelization factors **P** = 2, **P** = 4 and **P** = 8.

[2]Note that for comparison the trivial *Sphere* benchmark $f(x) = \sum_{i=1}^{6} x_i^2$ takes less than 0.0000001 Seconds on the considered CPU setup and is therefore around two to three magnitudes faster to evaluate than the Cassini1 benchmark.

[3]The potential speed up is calculated as average $Blocks$ for **P** = 1 divided by average $Blocks$ for considered **P**

[4]The potential speed up is calculated as average $Time$ for **P** = 1 divided by average $Time$ for considered **P**

TABLE III: Potential and actual speed up for different **P**

P	Potential Speed Up	Actual Speed Up
2	1.87	1.74
4	3.18	2.61
8	6.88	4.31

From Table III it can be seen that the actual speed up is well below the potential speed for **P**=2 and **P**=3, which is expected for reasons explained above. Furthermore it can be seen from Table III that the potential speed up is very efficient. In case of the actual speed up, a maximal speed up of 4.31 times could be achieved for the factor **P**=8. Even though this actual speed up remains well below the potential one of 6.88, it is still a significant improvement in regard to the total cpu-time required to solve the Cassini1 benchmark.

In addition to those speed ups reported in Table III it can be see in in Table II, that the average number of function evaluation is not greatly effected (increased) by the parallelization factor. If and for how large parallelization factors such behaviour could be observed would be an interesting question (see the Conclusions).

IV. Conclusions

A numerical investigation on the impact of parallelization on the optimization performance for MIDACO on the Cassini1 space trajectory benchmark has been presented. It could be shown, that the parallelization approach by MIDACO is very effective in regard to serial processed function evaluation (denoted as $Blocks$ in Section II and Section III, which were considered as the potential speed up. Due to the relatively cheap cpu-time cost of the Cassini1 benchmark, the actual speed was lower as the potential one. However, still significant speed up could be realized. As expected, the highest actual speed up could be realized by the highest parallelization factor **P**=8 and was around 4.31 times in regard to the serial cpu-runtime behaviour.

As the MIDACO software is suitable for massive parallelization, future research might focus on higher parallelization factors, which might be available for example in Cloud environments (see [6]) or by GPGPU programming (see [16]). In case the observed speed up behaviour in Section III does scale up for massive parallelization, significant reductions in the in the cpu-time required for space trajectory optimization could be realized by the here proposed approach.

Acknowledgement

We would like to express our gratitude to the Advanced Concept Team (ACT) for creating and publishing the GTOP benchmark set. Further we would like to thank the European Space Agency (ESA-ESTEC/Contract No. 21943/08/NL/ST) and EADS Astrium Ltd (Stevenage, UK) for their professional and financial support on the MIDACO development.

References

[1] Addis, B., Cassioli, A., Locatelli, M., Schoen, F., Global optimization for the design of space trajectories, Comput. Optim. Appl. 48(3), pp. 635-652, 2011.

[2] Munawar, A.: Redesigning Evolutionary Algorithms for Many-Core Processors Ph.D. Thesis, Graduate School of Information Science and Technology, Hokkaido University, Japan, 2011

[3] Gregoire, D, Dorronsoro, B., Bouvry, P.: New State-Of-The-Art Results For Cassini2 Global Trajectory Optimization Problem. *Acta Futura* 5, pp. 65-72, 2012. (2012)

[4] European Space Agency (ESA) and Advanced Concepts Team (ACT). Gtop database - global optimisation trajectory problems and solutions, Software available at http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html, 2013.

[5] Izzo, D.: Global Optimization and Space Pruning for Spacecraft Trajectory Design. *Spacecraft Trajectory Optimization* Conway, B. (Eds.), Cambridge University Press, pp.178-199, 2010.

[6] Hokkaido University: High Performance Computing System. Website available at http://www.hucc.hokudai.ac.jp/ (Japanese), 2014.

[7] Mike Dewar: NAG Blog - Reverse Communication. Available at http://blog.nag.com/2010/01/reverse-communication.html, 2010.

[8] K. Schittkowski, *NLPQLP - A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search (User Guide)*, Report, Department of Computer Science, University of Bayreuth, Bayreuth, 2009

[9] Schlueter, M., Egea, J.A., Banga, J.R.: Extended antcolony optimization for non-convex mixed integer nonlinear programming. Comput. Oper. Res. 36(7), 2217–2229 (2009)

[10] Schlueter, M., Gerdts, M.: The Oracle Penalty Method. J. Global Optim. 47(2), 293–325 (2010)

[11] Schlueter M., Gerdts M., Rueckmann J.J.: A Numerical Study of MIDACO on 100 MINLP Benchmarks. Optimization 61(7), 873–900 (2012)

[12] Schlueter M., Munetomo M.: Parallelization Strategies for Evolutionary Algorithms for MINLP. Proceedings of the IEEE Congress on Evolutionary Computation (IEEE-CEC), 635 – 641 (2013)

[13] Schlueter M., Erb S., Gerdts M., Kemble S., Rueckmann J.J.: MIDACO on MINLP Space Applications. Advances in Space Research, 51(7), pp. 1116-1131, 2013.

[14] Schlueter M.: MIDACO - Global Optimization Software for Mixed Integer Nonlinear Programming. Software available at http://www.midaco-solver.com, 2014

[15] Stracquadanio, G., Ferla, A., Felice, M., Nicosia, G.: Design of robust space trajectories. Research and Development in Intelligent Systems XXVIII, 341–354, Springer London, 2011.

[16] Michel Mueller: Typhoon Computing - Solutions for HPC. Software available at http://typhooncomputing.com/, 2014.

Appendix

Here the individual results of all 400 test runs corresponding to Table II are given in detail. Note that Table II summarizes the here displayed results in terms of average values.

Note that for the highest parallelization factor **P**=8 in two cases (test run Nr. 2 and Nr. 53) the cpu-time was less than 0.5 Seconds and is reported as zero in Table X, because the time was measured as integer and rounded to the next integer value.

TABLE IV: Results of 100 MIDACO test runs with factor **P**=1

Run	$f(x)$	$Blocks$	$Eval$	$Time$
1	4.977984	634563	634563	20
2	4.979983	1564283	1564283	50
3	4.979983	1560674	1560674	49
4	4.979761	996363	996363	31
5	4.977521	1470711	1470711	47
6	4.976461	698658	698658	22
7	4.977437	2096160	2096160	66
8	4.979034	782702	782702	25
9	4.979987	4856985	4856985	153
10	4.979858	1849866	1849866	58
11	4.968665	390831	390831	12
12	4.979537	2405631	2405631	77
13	4.972406	3000136	3000136	95
14	4.979983	1488174	1488174	47
15	4.978690	3159815	3159815	102
16	4.979224	2300385	2300385	78
17	4.979473	11786614	11786614	378
18	4.979983	1460200	1460200	46
19	4.972406	2847729	2847729	90
20	4.977124	9048186	9048186	283
21	4.978985	1944079	1944079	62
22	4.974628	625015	625015	19
23	4.975823	8590399	8590399	270
24	4.979996	810530	810530	26
25	4.979959	18469458	18469458	582
26	4.979964	7449400	7449400	234
27	4.979935	561689	561689	20
28	4.964064	2608602	2608602	83
29	4.979983	1193247	1193247	37
30	4.979994	1562439	1562439	49
31	4.979689	2537028	2537028	80
32	4.979963	804678	804678	26
33	4.976789	2218286	2218286	69
34	4.979696	4116111	4116111	130
35	4.979988	1658249	1658249	52
36	4.975766	1389870	1389870	44
37	4.977942	3692762	3692762	115
38	4.972406	2543443	2543443	80
39	4.977991	490550	490550	16
40	4.965957	1908810	1908810	60
41	4.980000	2020655	2020655	63
42	4.979846	11189586	11189586	352
43	4.963218	11226499	11226499	351
44	4.979963	678642	678642	21
45	4.972406	2214351	2214351	70
46	4.979983	1039529	1039529	33
47	4.979983	996306	996306	31
48	4.972406	2321249	2321249	73
49	4.979964	2285161	2285161	73
50	4.979722	2513016	2513016	79

TABLE V: continuation of Table IV

Run	$f(x)$	$Blocks$	$Eval$	$Time$
51	4.954630	150707	150707	5
52	4.979995	4124385	4124385	130
53	4.979983	915442	915442	29
54	4.978756	1831653	1831653	58
55	4.979959	17949198	17949198	560
56	4.979964	7208676	7208676	228
57	4.961516	3929263	3929263	123
58	4.979734	1383746	1383746	44
59	4.979983	823451	823451	26
60	4.979983	812499	812499	26
61	4.979417	1452461	1452461	45
62	4.979983	782138	782138	25
63	4.979983	768841	768841	24
64	4.979983	762085	762085	25
65	4.976705	2340658	2340658	73
66	4.975556	5365385	5365385	169
67	4.972406	2020084	2020084	64
68	4.978712	11389810	11389810	361
69	4.979878	4231856	4231856	134
70	4.974724	794829	794829	25
71	4.979974	7734391	7734391	242
72	4.979863	1765480	1765480	56
73	4.978351	967558	967558	30
74	4.978690	2383832	2383832	75
75	4.979634	951710	951710	31
76	4.979945	933795	933795	29
77	4.979903	1863150	1863150	59
78	4.977014	1142396	1142396	36
79	4.979903	1751927	1751927	56
80	4.979327	2662533	2662533	83
81	4.978690	2239531	2239531	71
82	4.977057	7272740	7272740	227
83	4.978822	1127953	1127953	35
84	4.969337	2090187	2090187	67
85	4.972406	1775203	1775203	59
86	4.978269	467810	467810	15
87	4.975477	2193450	2193450	69
88	4.979900	6094782	6094782	190
89	4.974309	1929402	1929402	61
90	4.979973	3082588	3082588	97
91	4.979991	11454753	11454753	359
92	4.979900	5964797	5964797	186
93	4.979903	1651414	1651414	57
94	4.979950	13044396	13044396	409
95	4.978598	2195714	2195714	69
96	4.977777	415837	415837	15
97	4.978018	1422327	1422327	51
98	4.977777	393412	393412	13
99	4.972406	1433660	1433660	45
100	4.972406	1563214	1563214	49
Average	4.977279	3150654	3150654	99

TABLE VI: Results of 100 MIDACO test runs with factor **P**=2

Run	$f(x)$	$Blocks$	$Eval$	$Time$
1	4.979626	3723189	7446378	125
2	4.979811	823985	1647970	29
3	4.977298	1309535	2619070	48
4	4.979799	3648436	7296872	121
5	4.977298	1290976	2581952	44
6	4.966056	3038267	6076534	102
7	4.979846	675180	1350360	23
8	4.976996	557607	1115214	19
9	4.979880	6101350	12202700	204
10	4.974892	3093822	6187644	102
11	4.976996	572843	1145686	19
12	4.978202	1131451	2262902	39
13	4.977298	1251422	2502844	41
14	4.979998	218417	436834	8
15	4.979990	11620840	23241680	387
16	4.971464	1328647	2657294	46
17	4.976073	3521386	7042772	118
18	4.975577	2067524	4135048	69
19	4.969733	614025	1228050	21
20	4.974039	501268	1002536	16
21	4.979424	1106061	2212122	38
22	4.978912	1884980	3769960	63
23	4.971760	3386629	6773258	114
24	4.979975	589672	1179344	20
25	4.979651	1409073	2818146	47
26	4.977581	2869706	5739412	98
27	4.979507	1294413	2588826	44
28	4.976996	433889	867778	15
29	4.979587	1623279	3246558	55
30	4.979417	2433451	4866902	81
31	4.980001	2029523	4059046	69
32	4.978123	4268357	8536714	143
33	4.979233	484752	969504	17
34	4.977714	2129517	4259034	72
35	4.979669	322493	644986	11
36	4.979781	4134746	8269492	137
37	4.979704	726053	1452106	24
38	4.977835	1966773	3933546	66
39	4.979921	898723	1797446	30
40	4.976903	1266382	2532764	44
41	4.977893	407239	814478	13
42	4.979271	3232605	6465210	110
43	4.976996	350835	701670	12
44	4.977641	666430	1332860	23
45	4.979924	595292	1190584	20
46	4.979951	1088631	2177262	37
47	4.974799	1061696	2123392	36
48	4.977274	274627	549254	9
49	4.979951	1069341	2138682	36
50	4.979661	354971	709942	13

TABLE VII: Continuation of Table VI

Run	$f(x)$	$Blocks$	$Eval$	$Time$
51	4.978162	3920970	7841940	131
52	4.979973	2052396	4104792	69
53	4.960803	321004	642008	11
54	4.979086	342840	685680	12
55	4.979910	3320904	6641808	112
56	4.975623	690574	1381148	24
57	4.980005	3360194	6720388	114
58	4.979951	1015462	2030924	34
59	4.979918	563052	1126104	19
60	4.977835	1816105	3632210	62
61	4.979999	3015752	6031504	103
62	4.975951	513609	1027218	17
63	4.979811	1099075	2198150	38
64	4.978982	768515	1537030	26
65	4.967622	287169	574338	9
66	4.979999	2977349	5954698	101
67	4.976951	1259014	2518028	42
68	4.978839	264759	529518	9
69	4.969801	158413	316826	6
70	4.979626	3387947	6775894	112
71	4.979252	2021042	4042084	68
72	4.972052	731549	1463098	24
73	4.979959	705007	1410014	24
74	4.979986	1157456	2314912	40
75	4.978982	712109	1424218	25
76	4.979045	1921115	3842230	64
77	4.975886	374254	748508	12
78	4.979651	1029597	2059194	35
79	4.967682	4036956	8073912	134
80	4.979926	2025497	4050994	68
81	4.979914	412101	824202	13
82	4.979574	321976	643952	11
83	4.978941	575479	1150958	19
84	4.979910	3193435	6386870	106
85	4.979982	1514681	3029362	50
86	4.979713	3741935	7483870	125
87	4.970984	873412	1746824	29
88	4.979567	2456899	4913798	81
89	4.979484	828254	1656508	27
90	4.979945	688771	1377542	23
91	4.978960	565252	1130504	19
92	4.969893	1604404	3208808	54
93	4.975888	2784096	5568192	91
94	4.979974	1167624	2335248	39
95	4.979792	845008	1690016	29
96	4.977131	1295013	2590026	43
97	4.979001	990827	1981654	34
98	4.979994	1413400	2826800	48
99	4.978966	2745732	5491464	93
100	4.969381	2999164	5998328	100
Average	4.977542	1683175	3366349	57

TABLE VIII: Results of 100 MIDACO test runs with factor **P**=4

Run	$f(x)$	$Blocks$	$Eval$	$Time$
1	4.968922	1431676	5726704	54
2	4.971511	833567	3334268	31
3	4.973130	1063314	4253256	40
4	4.977000	269367	1077468	10
5	4.960430	1502910	6011640	57
6	4.976738	398792	1595168	15
7	4.979103	879457	3517828	33
8	4.979873	681842	2727368	26
9	4.979569	368704	1474816	14
10	4.979970	427066	1708264	16
11	4.979131	1414875	5659500	53
12	4.979688	423417	1693668	16
13	4.976741	711161	2844644	26
14	4.978922	46636	186544	2
15	4.979097	2022189	8088756	78
16	4.979747	1366228	5464912	51
17	4.976741	695877	2783508	27
18	4.979084	470328	1881312	18
19	4.976741	690255	2761020	26
20	4.978605	904992	3619968	35
21	4.976741	685592	2742368	27
22	4.979869	817873	3271492	31
23	4.979961	306488	1225952	11
24	4.977321	170120	680480	7
25	4.977688	1765484	7061936	68
26	4.979676	5000549	20002196	187
27	4.973130	985374	3941496	37
28	4.973130	975521	3902084	38
29	4.979871	304398	1217592	12
30	4.977531	142290	569160	5
31	4.979263	2615295	10461180	97
32	4.975010	883368	3533472	34
33	4.974607	2602611	10410444	98
34	4.966905	799476	3197904	32
35	4.978406	1439354	5757416	55
36	4.977864	134178	536712	6
37	4.980004	2539428	10157712	98
38	4.978637	504270	2017080	19
39	4.978552	2132687	8530748	81
40	4.979065	2013011	8052044	78
41	4.973130	930710	3722840	36
42	4.979682	202715	810860	8
43	4.979688	310655	1242620	12
44	4.973130	926380	3705520	35
45	4.979311	769244	3076976	29
46	4.979073	1469810	5879240	55
47	4.979361	1468505	5874020	55
48	4.980001	377724	1510896	15
49	4.978368	3477869	13911476	134
50	4.978851	548068	2192272	21

TABLE IX: Continuation of Table VIII

Run	$f(x)$	$Blocks$	$Eval$	$Time$
51	4.972068	225980	903920	9
52	4.978406	1346624	5386496	51
53	4.977415	1213581	4854324	47
54	4.979745	2950371	11801484	111
55	4.979952	948251	3793004	37
56	4.979621	669042	2676168	27
57	4.980004	1345300	5381200	51
58	4.979786	1385440	5541760	53
59	4.979852	345289	1381156	13
60	4.972603	530454	2121816	20
61	4.979938	984685	3938740	37
62	4.979947	1445522	5782088	55
63	4.980003	786099	3144396	30
64	4.979949	565311	2261244	21
65	4.979115	2822195	11288780	105
66	4.980000	1368590	5474360	52
67	4.977688	1483076	5932304	55
68	4.979688	215739	862956	8
69	4.975184	264689	1058756	10
70	4.979514	278619	1114476	11
71	4.973130	835188	3340752	31
72	4.977688	1318261	5273044	51
73	4.978491	973650	3894600	36
74	4.973130	828223	3312892	32
75	4.979232	450134	1800536	17
76	4.972918	399605	1598420	15
77	4.979584	666744	2666976	26
78	4.979787	1410684	5642736	55
79	4.977073	1600335	6401340	61
80	4.979688	203148	812592	9
81	4.979317	117888	471552	5
82	4.978090	397275	1589100	14
83	4.979962	310188	1240752	13
84	4.976052	2147969	8591876	81
85	4.979965	818121	3272484	32
86	4.979988	1663266	6653064	64
87	4.973769	1074652	4298608	41
88	4.979786	1324276	5297104	51
89	4.979999	398502	1594008	16
90	4.978531	121201	484804	5
91	4.979899	741904	2967616	28
92	4.979688	156422	625688	6
93	4.976703	301461	1205844	11
94	4.973130	758712	3034848	29
95	4.979887	941007	3764028	35
96	4.979688	137513	550052	5
97	4.979912	1025244	4100976	39
98	4.979911	2105429	8421716	84
99	4.978349	416583	1666332	17
100	4.979139	427313	1709252	17
Average	4.977675	991475	3965898	38

TABLE X: Results of 100 MIDACO test runs with factor **P**=8

Run	$f(x)$	$Blocks$	$Eval$	$Time$
1	4.979992	296152	2369216	15
2	4.979447	18090	144720	0
3	4.979746	981382	7851056	50
4	4.974262	111520	892160	5
5	4.977147	185485	1483880	9
6	4.961017	147473	1179784	8
7	4.978063	17893	143144	1
8	4.967886	405032	3240256	20
9	4.980005	628804	5030432	31
10	4.979967	1227563	9820504	60
11	4.977202	242486	1939888	12
12	4.979381	1350673	10805384	66
13	4.979944	702115	5616920	35
14	4.974159	319526	2556208	15
15	4.961017	131462	1051696	7
16	4.974036	415935	3327480	21
17	4.979789	511485	4091880	25
18	4.979944	693228	5545824	34
19	4.979868	192582	1540656	9
20	4.979306	228889	1831112	12
21	4.979613	1161154	9289232	56
22	4.961017	118406	947248	6
23	4.971417	95302	762416	5
24	4.979946	93208	745664	5
25	4.979515	586578	4692624	28
26	4.979360	94583	756664	5
27	4.979360	94181	753448	5
28	4.976555	78922	631376	4
29	4.979985	94973	759784	4
30	4.961017	106502	852016	6
31	4.977316	245139	1961112	12
32	4.979828	59693	477544	3
33	4.979092	77389	619112	4
34	4.976318	330043	2640344	16
35	4.979564	247922	1983376	12
36	4.979984	44914	359312	3
37	4.979036	222371	1778968	11
38	4.979067	100949	807592	5
39	4.975697	199430	1595440	9
40	4.976327	218784	1750272	12
41	4.971829	20826	166608	1
42	4.968860	360179	2881432	18
43	4.979860	197694	1581552	9
44	4.979394	469150	3753200	23
45	4.978357	381507	3052056	20
46	4.979627	117283	938264	6
47	4.978478	135961	1087688	6
48	4.978799	474435	3795480	24
49	4.978677	336900	2695200	16
50	4.977714	1401823	11214584	69

TABLE XI: Continuation of Table X

Run	$f(x)$	$Blocks$	$Eval$	$Time$
51	4.978654	587892	4703136	29
52	4.979472	524771	4198168	25
53	4.979735	3679	29432	0
54	4.979958	48590	388720	3
55	4.976677	3310001	26480008	161
56	4.979652	2145672	17165376	104
57	4.979845	2265321	18122568	111
58	4.979467	245022	1960176	12
59	4.978962	285583	2284664	14
60	4.979886	943349	7546792	46
61	4.976318	261912	2095296	13
62	4.974684	235505	1884040	12
63	4.978243	107857	862856	5
64	4.979860	169262	1354096	8
65	4.979975	59168	473344	3
66	4.979931	240143	1921144	12
67	4.970576	74366	594928	4
68	4.971739	1421631	11373048	69
69	4.979433	549763	4398104	27
70	4.978824	467981	3743848	23
71	4.974357	294552	2356416	14
72	4.979358	15721	125768	1
73	4.975325	61921	495368	3
74	4.977317	1827550	14620400	90
75	4.979775	288764	2310112	14
76	4.978663	154960	1239680	8
77	4.980001	337983	2703864	17
78	4.979135	278835	2230680	13
79	4.972899	54674	437392	3
80	4.979965	1082869	8662952	53
81	4.979860	133399	1067192	7
82	4.979850	199851	1598808	10
83	4.979456	282213	2257704	13
84	4.978636	203299	1626392	11
85	4.978943	951734	7613872	46
86	4.979890	119626	957008	6
87	4.979795	674112	5392896	33
88	4.979009	206281	1650248	11
89	4.979715	614780	4918240	30
90	4.978799	414409	3315272	21
91	4.979278	118087	944696	5
92	4.979860	117163	937304	6
93	4.979996	368753	2950024	18
94	4.977104	782207	6257656	39
95	4.979746	763346	6106768	37
96	4.979961	185314	1482512	10
97	4.975756	210025	1680200	10
98	4.972595	354758	2838064	17
99	4.979150	3660198	29281584	179
100	4.973305	138900	1111200	7
Average	4.977332	458177	3665418	23