

A GLOBAL OPTIMISATION TOOLBOX FOR MASSIVELY PARALLEL ENGINEERING OPTIMISATION

Francesco Biscani, Dario Izzo, Chit Hong Yam

ABSTRACT

A software platform for global optimisation, called PaGMO, has been developed within the Advanced Concepts Team (ACT) at the European Space Agency, and was recently released as an open-source project.

PaGMO is built to tackle high-dimensional global optimisation problems, and it has been successfully used to find solutions to real-life engineering problems among which the preliminary design of interplanetary spacecraft trajectories - both chemical (including multiple flybys and deep-space maneuvers) and low-thrust (limited, at the moment, to single phase trajectories), the inverse design of nano-structured radiators, the design of non-reactive controllers for planetary rovers. Featuring an arsenal of global and local optimisation algorithms (including genetic algorithms, differential evolution, simulated annealing, particle swarm optimisation, compass search, improved harmony search, and various interfaces to libraries for local optimisation such as SNOPT, IPOPT, GSL and NLOpt), PaGMO is at its core a C++ library which employs an object-oriented architecture providing a clean and easily-extensible optimisation framework. Adoption of multi-threaded programming ensures the efficient exploitation of modern multi-core architectures and allows for a straightforward implementation of the island model paradigm, in which multiple populations of candidate solutions asynchronously exchange information in order to speed-up and improve the optimisation process. In addition to the C++ interface, PaGMO's capabilities are exposed to the high-level language Python, so that it is possible to easily use PaGMO in an interactive session and take advantage of the numerous scientific Python libraries available.

1. INTRODUCTION

With the introduction of mass-produced multi-core architectures, personal computers are becoming increasingly capable of performing parallel computations. Yet, the effort to parallelize algorithms is time consuming and often not attractive, especially

in scientific computing where software reuse is not as spread a practice as in other fields of computing. The open-source project PaGMO, (Parallel Global Multiobjective Optimiser), aims at filling this gap for optimisation algorithms providing, through a generalization of the so called island model (i.e. a coarse grained approach to parallelization of genetic algorithms) to all types of algorithms (population based and not), a simple experimenting platform that allows scientists to easily code algorithms and problems without having to care at all about the underlying parallelization that is provided 'for free' by the PaGMO infrastructure. The resulting software platform, participating to the Google initiative Summer of Code 2010, is described in this paper together with application examples to real life engineering problems of interest to aerospace engineers.

Recent results in global optimisation algorithms applied to the design of chemically-propelled interplanetary trajectories have shown how a straightforward application of off-the-shelf optimisation algorithms does not suffice to find satisfactory solutions for the most complex cases such as the Messenger trajectory or the Cassini or the TandEM trajectory [15]. While a wise use of the algorithms still provides useful information also in these most complex cases, the final optimal solutions need a substantial amount of engineering knowledge to be found. In this paper we show how the use of PaGMO allows different algorithms to cooperate to the solution of the same interplanetary trajectory problem, allowing to find solutions also in the most difficult cases in a reasonable time. In particular, we demonstrate a fully automated search of the solution space based on the use of Differential Evolution, Simulated Annealing and local search in a cooperative fashion. Information is exchanged asynchronously between the solvers operating in parallel CPUs via the implementation of a generalized migration operator offered by PaGMO. We test this search strategy in the case of the Cassini, TandEM and Messenger trajectories as defined in the European Space Agency Global Trajectory optimisation Problems database (GTOP) [17, 44]. We show that the algorithms are able to locate interesting regions of the search space and in particular to find the possible resonances. In the case of TandEM and Cassini, an automatic pruning strategy is able to successfully identify the best known solutions. In

the case of Messenger, the automated search locates a large number of possible solution clusters (due to the possible resonances at Mercury). A second run of the search focussed on a particular one of these clusters holds satisfactory results and is, in particular, able to find the same strategy adopted by the actual Messenger mission. A last example is then presented, where 4406 simpler interplanetary trajectories are optimised (each five times) taking advantage of PaGMO's parallelization capabilities in a reasonably short time to locate preliminarily good targets for an asteroid sample return mission in the 2020-2050 time frame.

2. PaGMO

PaGMO is an optimisation framework developed within the Advanced Concepts Team of the European Space Agency. Written in C++, PaGMO aims to provide an extensible infrastructure for defining optimisation problems (nonlinear, continuous, integer, mixed-integer, box-constrained, nonlinearly constrained, multi-objective optimisation is supported), coupled with a wide arsenal of global and local optimisation algorithms - some of them coded directly within PaGMO, others called from external libraries through thin wrappers. At the time of this writing, PaGMO provides the following optimisation algorithms:

- global and local optimisation algorithms coded directly within PaGMO, including a simple genetic algorithm [12], differential evolution [43], particle swarm optimisation [20], adaptive neighbourhood simulated annealing [7], improved harmony search [25], compass search [21], monotonic basin hopping [46], generalised multistart and Monte Carlo search [28];
- wrapper for SNOPT [11];
- wrapper for IPOPT [45];
- wrappers for algorithms from the NLOpt library [18], including Subplex (an extension of the classical Nelder-Mead method [35]), COBYLA [32] and BOBYQA [33];
- wrappers for algorithms from the GSL library [10], including the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [4], Fletcher-Reeves and Polak-Ribière nonlinear conjugate gradient methods [39] and the classical Nelder-Mead method [30];
- wrappers for algorithms from the SciPy library [19] (only available in the Python bindings), including fmin (Nelder-Mead), L-BFGS-B [48], sequential least-square programming [22] and truncated Newton method [29].

PaGMO provides automatic parallelisation of the optimisation process via a coarse-grained approach based on the island model [26], in which multiple optimisation instances of the same problem are launched at the same time, asynchronously exchanging information and improving the overall convergence properties of the optimisation. In PaGMO's implementation of the island model, each optimisation instance (i.e., each island) is launched in a separate thread of execution, thus automatically taking advantage of modern multiprocessor machines. The connections between islands are represented by a graph topology in which each node corresponds to an island and the edges represent routes through which candidate solutions can be communicated from one island to the other. The graph topologies can be either constructed by manually adding nodes and edges, or they can be selected among those already coded within PaGMO, including:

- fully connected topology;
- various types of ring topologies;
- small-world network topologies:
 - Barabási-Albert model [1];
 - Watts-Strogatz model [47];
- $G(n, p)$ Erdős-Rényi random graph [9];
- torus topology;
- cartwheel topology;
- lattice topology;
- hypercube topology;
- broadcast topology;
- wheel rim topology.

Some of the topologies available in PaGMO are visualised in Figure 1. Full control over the fine-grained details of the migration strategy (e.g., migration frequency and rate, selection and replacement policies) is provided. A preliminary study of the impact the topology on the optimisation process can be found in [36]. The class that contains the set of islands collaborating in an optimisation process, the topology and the migration policies is known in PaGMO as an *archipelago*.

PaGMO ships with a number of implemented optimisation problems readily available for use, such as:

- classical continuous test functions, such as Rastrigin, Rosenbrock [34], Schwefel, Griewank, Branin, Himmelblau, Lennard-Jones potential [23] and Levy5;
- constrained continuous test functions from [24];

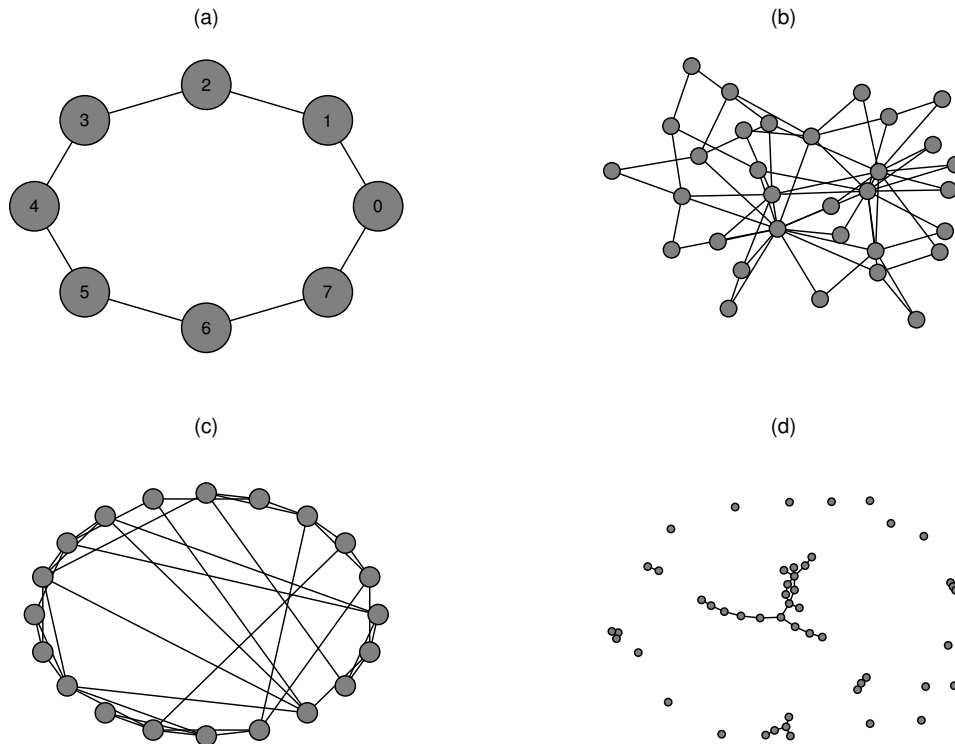


Figure 1: A selection of topologies available in PaGMO: ring topology (a), Barabási-Albert model (b), Watts-Strogatz model (c) and Erdős-Rényi $G(n, p)$ random graph (d).

- integer programming problems: Golomb ruler [40], 0-1 knapsack problem [27];
- multi-objective optimisation test problems from [8];
- all the chemical interplanetary spacecraft trajectory problems from the European Space Agency’s GTOP database [17, 44];
- an interplanetary multiple gravity assist low-thrust problem.

PaGMO’s C++ capabilities are exposed to the high-level language Python, so that it is possible to instantiate problems, algorithms, topologies and islands from either a script or an interactive Python session. It is also possible to define new problems and algorithms directly from Python, thus allowing on one hand to rapidly prototype and evaluate new ideas, and on the other to leverage the rich ecosystem of freely-available scientific Python modules (e.g., numerical integrators, machine learning libraries, computer algebra systems, etc.). Coupled with the matplotlib plotting module and the enhanced Python shell IPython, PaGMO’s Python bindings (which have been called PyGMO) offer a user-friendly interactive graphical experience.

3. SOME EXAMPLES

As an example of the use of PaGMO to solve engineering problems we report here the results of the application of the optimisation strategy described in the previous section to four trajectory optimisation selected problems. The first three problems are taken from the European Space Agency Global Trajectory optimisation (GTOP) database [17, 44]. The problems selected are among the most difficult proposed in the database and are included in the basic PaGMO distribution. They all are box-constrained, continuous, single objective optimisation problems, representing a multiple gravity assist interplanetary trajectory with one deep space maneuver allowed in each trajectory leg. The search space includes launch windows spanning decades (see the GTOP database for the precise definitions of the allowed bounds on the launch and fly-by dates). The fourth problem is a simpler problem admitting though a large number of different instances. We take advantage of PaGMO parallelization to find solutions to 4406 different instances of the problem in a reasonable computing time.

Experimental setup All the optimisation problems were set up in an archipelago of 5-7 islands

(depending on the number of available cores on the machine at the time of the experiment), equipped with a wheel rim topology (see Figure 2). The wheel rim topology consists of a classical bidirectional ring topology with an additional island at the center, fully connected to all the other islands. We chose to deploy global optimisation algorithms (namely, adaptive neighbourhood simulated annealing from [7] and differential evolution from [43]) on the ring, and a local optimisation algorithm (namely, the Subplex algorithm from [35] as implemented in [18]) in the center.

The motivations behind these choices are the following:

- the ring topology is a proven and popular choice in the context of parallel population-based algorithms, as shown for instance in [14, 42, 13, 5, 6, 16, 2];
- the additional island in the center receives through migration the best results of the global optimisation algorithms in the ring, refines them through a local search, and migrates them back to the ring. Its role is hence, on one hand, to improve the results of the global search, and on the other to inject back diversified candidate solutions into the ring;
- regarding the choice of the algorithms, both simulated annealing and differential evolution have proven to be effective for the optimisation of interplanetary spacecraft trajectories (as shown for instance in [16]), whereas the derivative-free Subplex method, a refinement of the classical Nelder-Mead algorithm, is particularly suited for the noisy and multi-modal objective functions appearing in these optimisation problems.

In order to give an example of use of PaGMO, we reproduce here the Python code necessary to perform one optimisation run with the setup described above:

```

1 # Import the PyGMO classes
2 from PyGMO import *
3
4 # Instantiate the algorithms
5 sa = algorithm.sa_corana(10000,1,0.01)
6 de = algorithm.de(500,0.8,0.9)
7 local = algorithm.nlopt_sbplx(500,1e-4)
8
9 # Instantiate the problem
10 prob = problem.messenger_full()
11
12 # Build the archipelago
13 a = archipelago(topology.rim())
14 a.push_back(island(prob,local,1,1.0,
15 migration.worst_r_policy()))
16 a.push_back(island(prob,sa,1))
17 a.push_back(island(prob,de,20))
18 a.push_back(island(prob,sa,1))
19 a.push_back(island(prob,de,20))
20 a.push_back(island(prob,sa,1))
21 a.push_back(island(prob,de,20))

```

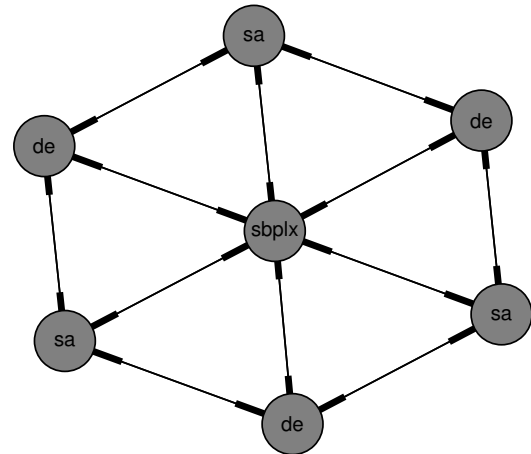


Figure 2: Experimental setup: an archipelago with wheel rim topology, global optimisation algorithms on the outer ring (simulated annealing and differential evolution) and a local optimisation algorithm in the inner island (Subplex).

```

21
22 # Perform evolution twenty times
23 a.evolve(20)
24 a.join()

```

Detailed explanation:

- on line 2, all the PaGMO classes are imported into the current namespace;
- on lines 5-7, the algorithms are instantiated;
- on line 10, the problem (in this case the full Messenger problem) is instantiated;
- on lines 13-20, the archipelago is instantiated:
 - on line 13, an empty archipelago with rim topology is created;
 - on line 14, the central island¹ is created and inserted in the archipelago with the `push_back()` method. The island is constructed from the problem `prob` and the algorithm `local`, it contains one single individual², has a probability of accepting the migrating individuals of 100% and replacement policy `migration.worst_r_policy()`, which

¹By convention, in the rim topology the first island inserted is the central one.

²In general, local optimisation algorithms in PaGMO will not benefit from multiple individuals in the population, and their default behaviour is to optimise just the best individual in the population.

will unconditionally replace the worst individual in the island with the incoming individuals. This island needs a non-default replacement policy because we want it to optimise *every* candidate solution coming from the ring, whereas the default behaviour would be to accept migrating individuals only if they would improve upon the worst individuals present in the population (which is the behaviour frequently desired for population-based algorithms);

- on lines 15-20, the ring islands are created and inserted into the archipelago. The simulated annealing islands operate on populations of a single individual, whereas the differential evolution islands are instantiated with a population of 20 individuals. The default migration policies are in these cases appropriate;
- on line 23, the optimisation process is started by calling the `evolve()` method of the archipelago. The argument passed to the `evolve()` method, in this case 20, means that each algorithm on each island is called 20 times with the parameters passed in the constructors on lines 5-7. E.g., in case of differential evolution, it means that the algorithm is run for $500 \cdot 20 = 10000$ generations, with weight coefficient equal to 0.8 and crossover probability equal to 0.9. Migration is allowed to happen at the end of each one of the 20 internal iterations of the `evolve()` method;
- on line 24, the archipelago is joined, meaning that the flow of the program stops until the optimisation run started on line 23 has concluded. Since PaGMO runs asynchronously each algorithm in a separate thread, the `evolve()` call on line 23 will return almost immediately – the optimisation process having forked in the background. The `join()` call blocks the program until the optimisation has finished.

Optimisation strategy For the first three problems we adopted the following optimisation strategy:

1. we instantiated an archipelago with rim topology as described above and let the optimisation run for an amount of time variable from 2 to 5 minutes;
2. at the end of each optimisation run, we recorded the best candidate solution produced and reset the archipelago with randomly chosen initial conditions.

Step 1. and 2. were automatically repeated for a statistically significant number of times (usually in the order of hundreds), thus producing a collection of optimised candidate solutions. This collection of

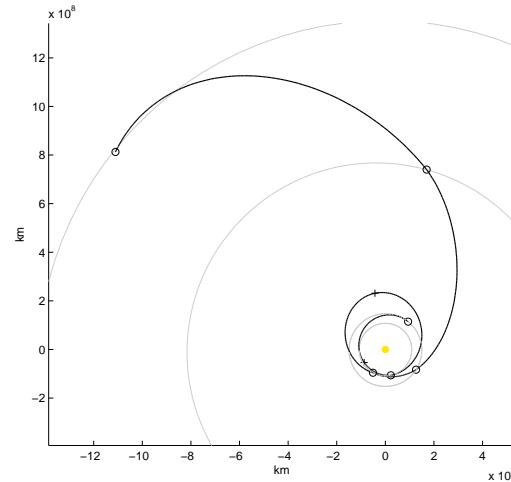


Figure 3: Visualization of the found trajectory for `cassini_2`.

candidate solutions was then analysed with the clustering algorithm described in [15], which shrinks the search bounds of each variable of the decision vector to the bounds containing the top candidate solutions in terms of objective function values (in this specific instance, we chose the top 20% candidates). The new bounds are then used to launch another round of multistart optimisations. This cycle can be repeated multiple times (e.g., until the bounds are not shrunked appreciably any more).

For the sample return problem, which involves the solution of multiple optimisation problems considerably easier than the other problems considered in the tests, there was no need to perform the bounds pruning iterations. A single run of the optimisation algorithms in the archipelago was usually enough to solve the individual problems.

3.1. Results on problem::cassini_2

This problem represents the interplanetary trajectory of the spacecraft Cassini. For a detailed description of this global optimisation problem we refer the reader to the GTOP database [17, 44]. For the purpose of this paper we just mention that the objective function represents the sum of all ΔV , including the launch, where the last contribution (Jupiter arrival) is relative velocity with respect to Jupiter at arrival. For this problem we apply the fully automated search described above with three pruning cycles. At the end of the process (employing 7CPUs for a period of roughly 8 hours) the best trajectory found is visualized in Figure 3. Details on the trajectory as reported in Table 1a. The trajectory is, essentially, the same best result posted in the database (22th May 2009) and found by M. Schlüeter, J. Fiala, M.

| | | | | | |
|-----------------------------|------------|------------------------------|------------|------------------------------|------------|
| Departure | | Departure | | Departure | |
| Epoch | 12/11/1997 | Epoch | 15/11/2021 | Epoch | 14/08/2005 |
| V_∞ | 3.254 km/s | V_∞ | 3.34 km/s | V_∞ | 3.95 km/s |
| Cruise | | Spacecraft | | Cruise | |
| DSM ΔV | 484 m/s | Departure mass | 2085.44 kg | Venus fly-by | 20/10/2006 |
| Venus fly-by | 29/04/1998 | Arrival mass | 1476.03 kg | DSM ΔV | 343 m/s |
| DSM ΔV | 399 m/s | I_{sp} | 312 s | Venus fly-by | 04/06/2007 |
| Venus fly-by | 27/06/1999 | Cruise | | DSM ΔV | 567 m/s |
| Earth fly-by | 19/08/1999 | Venus fly-by | 30/04/2022 | Mercury fly-by | 12/01/2008 |
| Jupiter fly-by | 31/03/2001 | Earth fly-by | 04/04/2023 | DSM ΔV | 91 m/s |
| Arrival | | DSM ΔV | 167 m/s | Mercury fly-by | 04/10/2008 |
| Epoch | 05/2007 | Earth fly-by | 25/06/2026 | DSM ΔV | 224 m/s |
| V_∞ | 4.25 km/s | Arrival | | Mercury fly-by | 28/09/2009 |
| Total flight time 9.4 years | | Epoch | 03/07/2031 | DSM ΔV | 179 m/s |
| (a) problem::cassini_2. | | V_{SOI} | 0.676 km/s | Arrival | |
| | | Total flight time 9.63 years | | Epoch | 03/2011 |
| | | | | V_{MOI} | 0.905 km/s |
| | | | | Total flight time 5.59 years | |
| | | | | (c) problem::messenger_full. | |
| | | (b) problem::tandem(6,10). | | | |

Table 1: Trajectory details for the best trajectory found for the first three interplanetary trajectory problems.

Gerdtts at the University of Birmingham using the MIDACO solver developed within the project “Non-linear mixed-integer-based Optimisation Technique for Space Applications” co-funded by ESA Networking Partnership Initiative, Astrium Limited (Stevenage, UK) and the School of Mathematics, University of Birmingham, UK [38]. The solution employs two deep space maneuvers during the first two legs as detailed in Table 1a.

3.2. Results on problem::tandem(6,10)

TandEM is one of the L-class candidate missions that were proposed in response to the European Space Agency call for proposals for the 2015-2025 Cosmic-Vision programme. Initially, the mission included a Titan orbiter and an Enceladus penetrator. The interplanetary part of the trajectory was preliminarily studied in 2008 by a Mission Analysis Outer Planets Working Group that included different experts from academia and space industry. In that preliminary study a baseline for the TandEM mission was defined and forms the basis of the problem here solved in an automated fashion using PaGMO. The baseline considers a launch with Atlas 501, and an injection orbit at Saturn with $e = 0.985$, $r_p = 80330$ km. The mission objective is to maximize the final spacecraft mass at arrival and to complete the trajectory within ten years. For a detailed description of this global op-

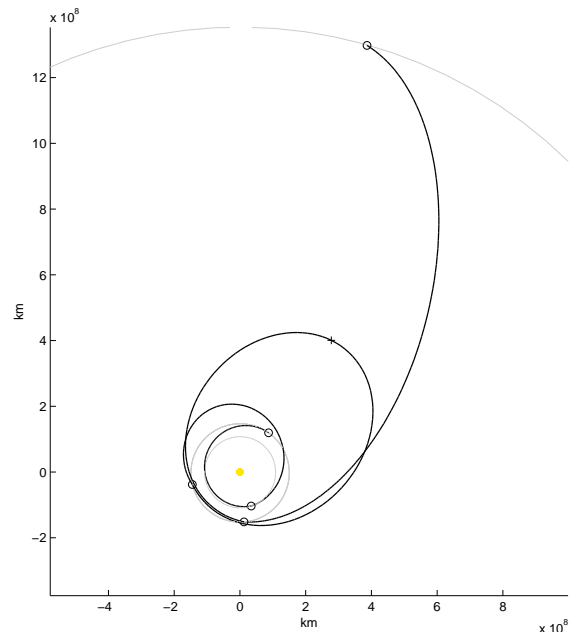


Figure 4: Visualization of the found trajectory for TandEM.

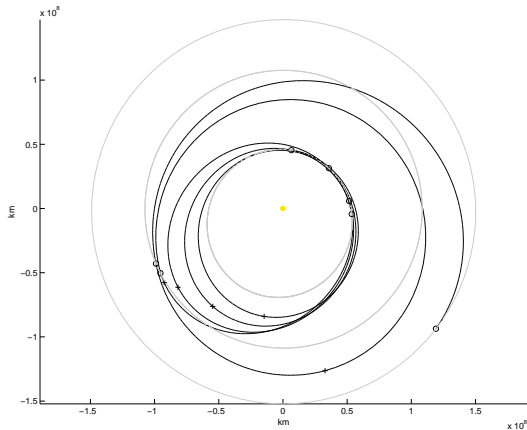


Figure 6: Visualization of the trajectory found for Messenger.

timisation problem we refer the reader to the GTOP database [17, 44]. For this problem we apply the fully automated search described above with three pruning cycles. At the end of the process (employing 7CPUs for a period of roughly 6 hours) the best trajectory found is visualized in Figure 4. Details on the trajectory are as reported in Table 1b. The solution found improves the previous best found by B. Addis, A. Cassioli, M. Locatelli, F. Schoen (from the Global Optimisation Laboratory, University of Florence) who also have the record on the solutions for all other TandEM problem instances (i.e. for different fly-by and time constraint). The final trajectory employs one only deep space manoeuvre during the Venus-Venus leg.

3.3. Results on problem::messenger_full

This problem is probably the most complex problem in the GTOP database and represents one of the most complex chemical interplanetary trajectory ever designed and flown, that of the Messenger spacecraft. Messenger, at the time of writing, is on its way to Mercury, where (roughly next year, in 2011) will become the first spacecraft to ever orbit around the planet. Its path in the solar system to reach its final destination included a long planetary tour: Earth-Earth-Venus-Venus-Mercury-Mercury-Mercury-Mercury. In the PaGMO version of the problem, the Messenger trajectory is transcribed into a box-constrained global optimization problem. The objective function is the total ΔV accumulated from the Earth launch to a Mercury Orbit Insertion (MOI) into an orbit having $e = 0.704$, $r_p = 2640$ km. For a detailed description of this global optimisation problem we refer the reader to the GTOP database [17, 44]. In this case, after a first run of the algorithm, cluster detection shows the presence of many different solution clusters re-

lated to the different possible resonances at Mercury, as shown in Figure 5. The best solution after the first algorithm run is around 5.15 km/sec. By focussing the optimization into one of the detected clusters we obtain a solution at around 2.3 km/sec which is lowering the GTOP database record substantially and is detailed in Table 1c and visualized in Figure 6.

3.4. Results on problem::sample_return

A single instance of this problem represents an interplanetary trajectory starting from the Earth and performing a rendezvous with a selected asteroid. After a minimum waiting time the spacecraft is required to come back to the Earth with a maximum hyperbolic encounter velocity. One deep-space maneuver per leg was allowed, creating a global optimisation problem of dimension 12. This type of trajectory can be used to perform the preliminary selection of possible final asteroids for sample return missions. The same trajectory model is also relevant to design human missions to asteroids. For the purpose of this paper we do not enter into the details on the system design and launch window choice, instead it is our interest to ‘just pick an example’ and show the possibility of using PaGMO to solve in a reasonable time a large number of problem instances (e.g. varying the final asteroid). We took all the asteroids listed in the JPL NEA database:

http://neo.jpl.nasa.gov/cgi-bin/neo_elem

having $H < 22$, which roughly corresponds to asteroids having diameter larger than 200m. For the selected 4406 asteroids we optimised the trajectory considering a launch in the 2020-2050 time frame, allowing a final encounter velocity at the Earth return of 4.5 km/sec and minimizing the total ΔV as evaluated from:

$$\Delta V = \Delta V_L + \Delta V_{dsm_1} + \Delta V_R + \Delta V_D + \Delta V_{dsm_2} + \Delta V_E,$$

where ΔV_L is the hyperbolic velocity when leaving the Earth sphere of influence, ΔV_{dsm_1} is the first deep space maneuver, ΔV_R is the rendezvous velocity, ΔV_D is the relative velocity at asteroid departure, ΔV_{dsm_2} is the second deep space maneuver and ΔV_E is the final braking maneuver to reduce the entry speed to 4.5 km/sec. A minimum waiting time on the asteroid of 5 days is also considered.

The computations were performed on an Xserve with 8 processing units and lasted 8 hours. The results are visualized in Figure 7.

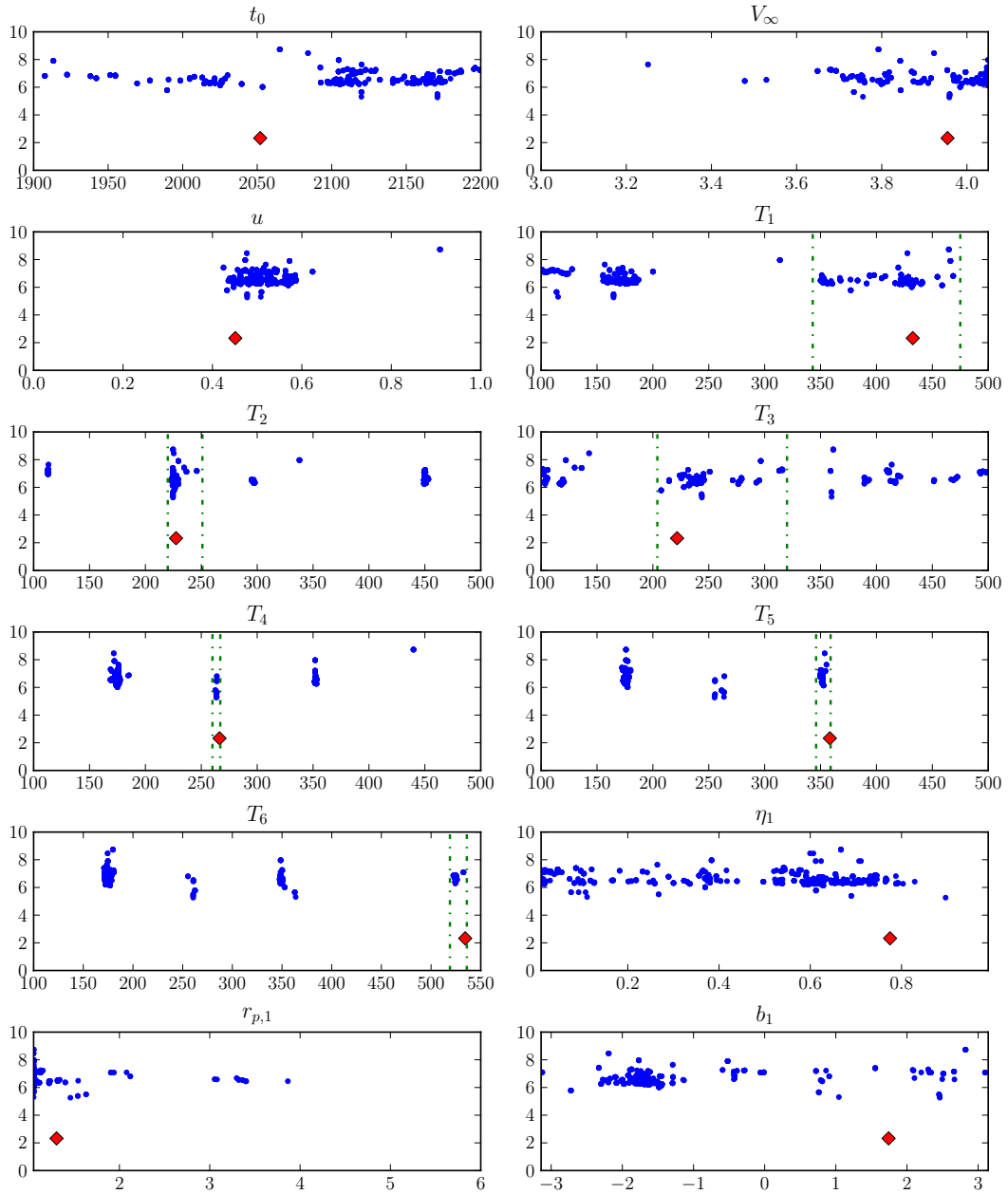


Figure 5: Manual pruning for a selection of variables from the Messenger problem. The blue dots represent the best candidate solutions of each run of the multistart strategy, the vertical green bars denote the new, narrowed bounds of the problem and the red diamond marker represents the final solution. The clustering of the best candidate solutions in correspondence with the resonant flybys at Mercury is clearly visible for the variables T_2 , T_4 , T_5 and T_6 .

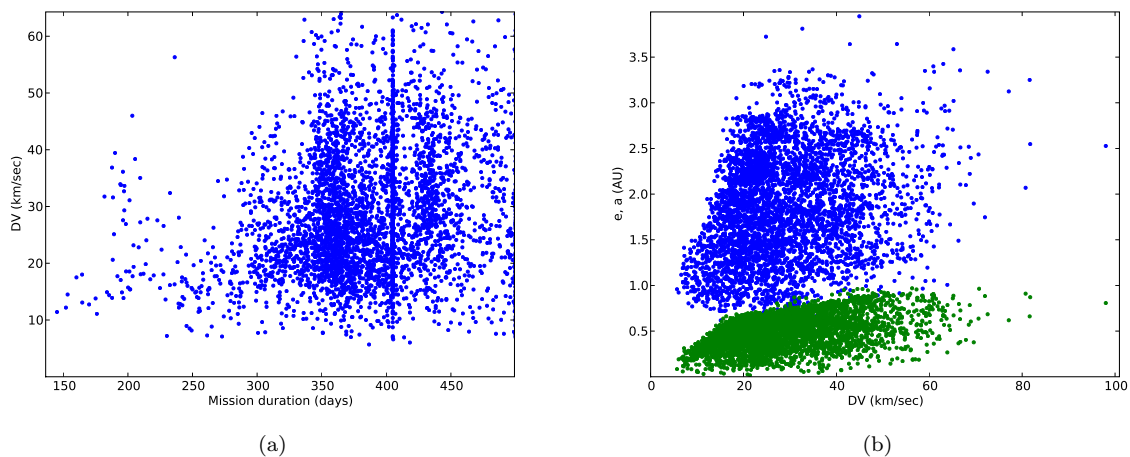


Figure 7: Relations between ΔV and properties of the best solutions found: mission duration against ΔV (a) and ΔV against semi-major axis and eccentricity (b).

4. CONCLUSIONS AND FUTURE WORK

In this paper we have presented PaGMO, a global optimisation software framework for parallel engineering optimisation developed within the Advanced Concepts Team at the European Space Agency. We have tested PaGMO on three hard, realistic interplanetary trajectory optimisation problems (TandEM, Cassini and Messenger), showing how PaGMO is able to find automatically the best known solutions for all three problems. With the help of a human-guided final pruning step, PaGMO was also able to locate the ‘real’ trajectory for the Messenger probe, consisting of multiple resonant flybys at Mercury. A fourth benchmark problem, consisting in the optimisation of simple trajectories to a selection of 4406 near-Earth asteroids, was shown with the intent of highlighting PaGMO’s parallelisation capabilities.

Future work on PaGMO will concentrate on areas such as:

- extension of the computational capabilities via interfacing to popular massively parallel frameworks, such as MPI [41] for scientific clusters and BOINC [3] for distributed computing;
- exploration of the possibility of using GPGPU computing [31] to speed-up the most time-consuming parts of the optimisation process;
- implementing/interfacing additional optimisation algorithms;
- interfacing with machine learning packages (such as PyBrain [37]), for easy coding of artificial intelligence problems.

Some of these activities will be tackled within the Google Summer of Code 2010, in which an international group of University students will be working during the summer on PaGMO under the mentorship of the PaGMO development team - while being sponsored by Google.

PaGMO is Free Software, and it is available for download from the SourceForge website:

<http://pagmo.sourceforge.net>

REFERENCES

- [1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, January 2002.
- [2] C. Ampatzis, D. Izzo, M. Ruciński, and F. Biscani. ALife in the Galapagos: migration effects on neuro-controller design. In *Proceedings of the European Conference on Artificial Life (ECAL 2009)*, September 2009.
- [3] D. P. Anderson. BOINC: A System for Public-Resource Computing and Storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, PA (USA), 2004. IEEE Computer Society.
- [4] C. G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms - 1. General Considerations. *IMA Journal of Applied Mathematics*, 6:76–90, 1970.
- [5] E. Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.
- [6] E. Cantú-Paz and M. Mejía-Olvera. Experimental results in distributed genetic algorithms. In

- International Symposium on Applied Corporate Computing*, pages 99–108, 1994.
- [7] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the “simulated annealing” algorithm. *ACM Transactions on Mathematical Software*, 13(3):262–280, September 1987.
- [8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.
- [9] P. Erdős and A. Rényi. On Random Graphs. I. *Publicationes Mathematicae*, 6:290–297, 1959.
- [10] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi. *GNU Scientific Library Reference Manual*. Network Theory Ltd, third edition, January 2009.
- [11] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47:99, 2005.
- [12] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
- [13] V. S. Gordon and L. D. Whitley. Serial and parallel genetic algorithms as function optimizers. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 177–183, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.
- [14] H. Homayounfar, S. Areibi, and F. Wang. An advanced island based GA for optimization problems. *Dynamics of continuous discrete and impulsive systems-series b-applications & algorithms*, pages 46–51, 2003.
- [15] D. Izzo. *Spacecraft Trajectory Optimization*, chapter 7. Cambridge Press, 2010.
- [16] D. Izzo, M. Ruciński, and C. Ampatzis. Parallel global optimisation meta-heuristics using an asynchronous island-model. In *2009 IEEE Congress on Evolutionary Computation*, pages 2301–2308. 2009 IEEE Congress on Evolutionary Computation (IEEE CEC 2009), Trondheim, Norway, May 18–21., 2009.
- [17] D. Izzo, T. Vinkó, and M. del Rey Zapatero. GTOP Database: Global Optimisation Trajectory Problems and Solutions. <http://www.esa.int/gsp/ACT/inf/op/globopt.htm>, 2010.
- [18] S. G. Johnson. The NLopt nonlinear-optimization package. <http://ab-initio.mit.edu/nlopt>, 2010.
- [19] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python. <http://www.scipy.org/>, 2010.
- [20] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Piscataway, NJ (USA), 1995.
- [21] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45:385, 2003.
- [22] D. Kraft. Algorithm 733: TOMP – Fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software*, 20:262–281, September 1994.
- [23] J. E. Lennard-Jones. On the Determination of Molecular Fields. II. From the Equation of State of a Gas. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character (1905-1934)*, 106:463–477, October 1924.
- [24] L. Lukšan and J. Vlček. Sparse and partially separable test problems for unconstrained and equality constrained optimization. Technical Report V-767, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague (Czech Republic), 1999.
- [25] M. Mahdavi, M. Fesanghary, and E. Daman-gir. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188:1567–1579, May 2007.
- [26] B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 428–433, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
- [27] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., New York, NY (USA), 1990.
- [28] N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44:335, September 1949.
- [29] S. G. Nash. A survey of truncated-Newton methods. *Journal of Computational and Applied Mathematics*, 124:45–59, December 2000.
- [30] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 00/1965 1965.
- [31] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.
- [32] M. J. D. Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7:287–336, 1998.
- [33] M. J. D. Powell. The BOBYQA algorithm for bound constrained optimization without deriva-

- tives. Technical Report NA2009/06, Department of Applied Mathematics and Theoretical Physics, Cambridge England, 2009.
- [34] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3:175–184, March 1960.
- [35] T. H. Rowan. *Functional stability analysis of numerical algorithms*. PhD thesis, University of Texas at Austin, Austin, TX (USA), 1990.
- [36] M. Ruciński, D. Izzo, and F. Biscani. On the Impact of the Migration Topology on the Island Model. *Accepted in Parallel Computing*, 2010.
- [37] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 2010.
- [38] M. Schlüter, J. A. Egea, L. T. Antelo, A. A. Alonso, and J. R. Banga. An Extended Ant Colony Optimization Algorithm for Integrated Process and Control System Design. *Industrial & Engineering Chemistry Research*, 48:6723–6738, July 2009.
- [39] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical Report CS-94-125, Carnegie Mellon University, Pittsburgh, PA (USA), 1994.
- [40] S. Sidon. Ein Satz über trigonometrische Polynome und seine Anwendung in der Theorie der Fourier-Reihen. *Mathematische Annalen*, 106:536–539, December 1932.
- [41] J. M. Squyres. Definitions and fundamentals – the message passing interface (MPI). *ClusterWorld Magazine, MPI Mechanic Column*, 1(1):26–29, December 2003.
- [42] T. Starkweather, L. D. Whitley, and K. E. Mathias. Optimization using distributed genetic algorithms. In *PPSN I: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 176–185, London, UK, 1991. Springer-Verlag.
- [43] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, November 1997.
- [44] T. Vinkó and D. Izzo. Global optimisation heuristics and test problems for preliminary spacecraft trajectory design. Technical Report GOHTPPSTD, European Space Agency, the Advanced Concepts Team, 2008.
- [45] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, March 2006.
- [46] D. J. Wales and J. P. K. Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101:5111–5116, July 1997.
- [47] D. J. Watts and S. H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393:440–442, June 1998.
- [48] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23:550–560, December 1997.