Multi-Objective Parameter Tuning for PSO-based Point Cloud Localization

Roberto Ugolotti and Stefano Cagnoni

Department of Information Engineering, University of Parma,
Parco Area delle Scienze 181/A, 43124 Parma, Italy

Abstract. It has been largely proven that population-based metaheuristics such as Particle Swarm Optimization (PSO) are severely affected by the choice of their parameters.
In this paper, we use a multi-objective parameter tuning method called EMOPaT (Evolutionary Multi-Objective Parameter Tuning) to optimize PSO when dealing with a real-world optimization task: the localization of an object acquired by a laser scanner in the form of a point cloud.
We want to optimize both the time needed to reach a quality threshold and the final alignment between the point cloud and a reference model of the object. Our system is able to generate "fast" and "precise" versions of PSO and, among all the possible configurations which lie between the fastest and the most precise, the ones that give the best trade-offs between precision and speed.

Keywords: Particle Swarm Optimization, Pattern Recognition, Multi-Objective Optimization, Point Clouds

1 Introduction

Evolutionary Computation (EC) and Swarm Intelligence techniques have proven to be very successful in several fields. Among the many possible applications, we have shown that these techniques are able to recognize and localize object in images and video sequences [15].

In a previous work [14], we compared the performance of GPU-based implementations of Particle Swarm Optimization (PSO [5]) and Differential Evolution (DE) [12] against Fast Point Feature Histograms (FPFH [11]) in recognizing and localizing an object acquired with a laser scanner in the form of a point cloud. The two metaheuristics proved their effectiveness, especially when dealing with noisy or incomplete data.

In this paper, we try to further improve the results obtained by PSO by using a multi-objective evolutionary method for automatic parameter tuning. Automatic parameter tuning is a procedure in which techniques such as EC-based metaheuristics are employed to automatically set good parameter values to configure a given algorithm. This procedure is especially useful in EC, because usually evolutionary metaheuristics have many parameters which, if not chosen

appropriately, may lead to bad results even on simple problems. Many different techniques have been proposed over the years such as hyper-heuristics [2], racing algorithms [7], model-based methods [1] and Meta-Evolutionary Algorithms [4]. Our multi-objective extension of this paradigm has been firstly presented in [13] and is summarized in Section 4.

The remainder of the paper is structured as follows. Section 2 presents the general framework within which this work has been developed; Section 3 introduces the problem under consideration; Section 4 describes the evolutionary method for parameter tuning; Sections 5 and 6 describe the experimental setup and results and are followed by some final remarks in Section 7.

2 Model Based Object Recognition

The approach used in this work to align two point clouds is an application of the general method described in [15], whose goal is to locate objects in static images or track them in video sequences. The general process is based on these steps:

1. A template of the object to recognize is created off-line, and the available range of deformations to which it can be subject is defined. A template is a mathematical description of the object and of all the possible ways in which it may appear in an image;
2. Every time a new image (or frame, in case of a video) is presented in input, the template is rotated and deformed according to the transformations represented by the individuals of a population based metaheuristic, such as PSO, driven by a fitness function that appropriately describe the match between the model and the target object under consideration;
3. The process stops when the alignment is good enough or the time allowed for performing the task has expired.

In the case presented here, we want to match the pose of a known object (which is part of a database of available models) with a point cloud acquired by a linear scanner, i.e., to find the transformation of the model which best matches the data in the point cloud. Therefore, our template is a point cloud that can only be subject to a rigid transformation, so the search space can be defined only by six degrees of freedom (translations and rotations around the three axes). Figure 1 shows an example of a PSO particle encoding a roto-translation.

Fig. 1. A PSO particle encoding a possible roto-translation of the reference.

More details regarding the specific application considered in this work are presented in the next section.

3 Point Cloud Alignment

Our method is designed for inclusion in an architecture whose goal is to help users to program robotic tasks, dealing with object handling. To reach this goal, a subsystem for object recognition has been developed (see Figure 2). It receives input data from a high-resolution planar laser scanner mounted on the wrist of a six degrees of freedom robot arm. The estimated accuracy of the whole measurement chain is about 1.5 cm. Data undergo several preprocessing steps that refine the acquisition and are then passed to the PSO-based recognizer, along with a list of models stored in a database. The output of the recognizer indicates which objects are present in the scene and in which pose. A more detailed description of a preliminary version of this system is presented in [10]. In this paper, we will only consider a single object, since our goal here is just to determine the pose of a known object as a real-world task on which EMOPaT's performance can be demonstrated.

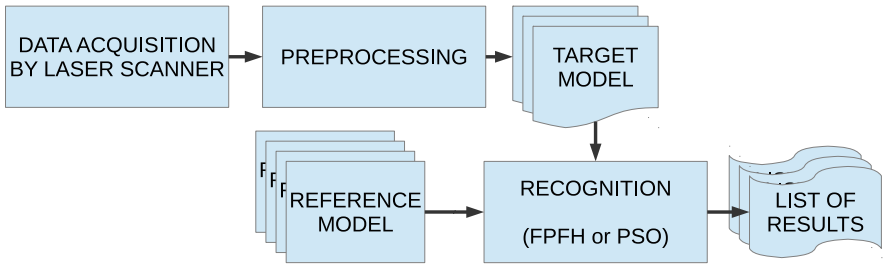

Fig. 2. Representation of the system within which the PSO recognizer is used.

The data acquired from the laser scanner is in the form of a point cloud, i.e., a set of three-dimensional points expressed within a certain coordinate system that represents the external surface of an object. Many solutions have been proposed for Point-cloud registration and alignment. Among others, Li et al [6] propose a function based on a Gaussian Mixture distance map optimized by PSO; in [8], registration of partially overlapped point clouds is achieved by estimating their Extended Gaussian Images; and in [16] the relative distance between a triangular mesh and a point cloud is minimized by means of DE.

3.1 Fitness Function

The goal of the task under consideration is to match the newly acquired point cloud (called target) to a reference one. If the two point clouds refer to the same object, which is the case we are considering in this paper, this is just an alignment problem and the fitness function to be minimized takes into consideration the distance of each point of the target to any of the points that compose the reference. If more models are considered, the same procedure can be used for

object classification. Within such a framework, the point cloud can be classified as the object in the database for which this fitness is minimal.

The fitness function used by PSO is relatively straightforward. We compare the target cloud T to be recognized (composed of N_T points), with the reference cloud R, composed of N_R points. This reference is subject to a transformation M encoded by a PSO particle (as in Figure 1), to obtain a roto-translated version $R' = M(R)$. The fitness of a particle is the average of the minimum distances of each point of T to the closest point of the roto-translated reference R'. More formally:

$$F(T, R') = \frac{1}{N_T} \sum_{p \in T} \min_{q \in R'} \left(dist(p, q) \right)$$

where $dist(x, y)$ is a valid distance metric between two points x and y; in this case we selected the squared euclidean distance.

Each point cloud is expressed within a local reference frame centered around its centroid. A model can do a full rotation around each axis while the range of translation is limited to 10 cm in each direction, which is good enough to satisfy the requirements of the environment we are considering.

4 EMOPaT

Evolutionary Multi-Objective Parameter Tuning (EMOPaT) was firstly presented in [13]. EMOPaT is a Meta-Evolutionary Algorithm (Meta-EA, see a representation in Figure 3), i.e. a method in which an Evolutionary Algorithm (in our case NSGA-II [3]) is not used directly to solve a problem, but to tune another metaheuristic (that can be an EA itself), which will then be used to solve the problem.

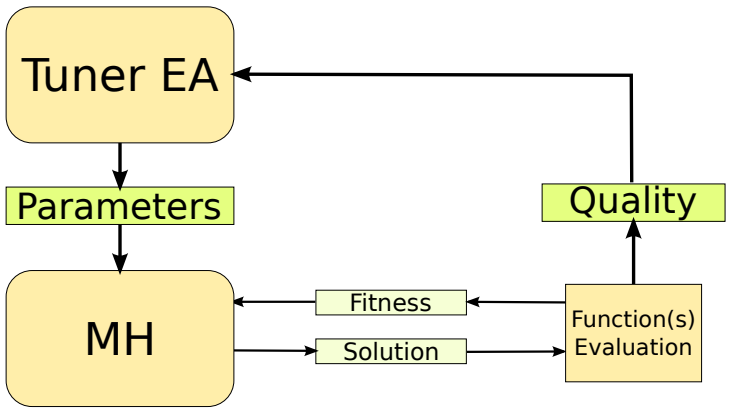

Fig. 3. Scheme of a general Meta-EA, the paradigm on which our tuning method relies.

A Meta-EA (see [4] for a detailed review) works similarly to any other EC technique. It generates a population of possible solutions, each of which is actually an instance of the metaheuristic to optimize. Therefore the tuner EA does not operate in the search space of the problem at hand, but in the search space of the parameters of the metaheuristic considered, which in turn will try to solve the problem.

A Meta-EA tries to find a good set of parameters according to a single objective, usually the fitness function of the EA to be tuned. It is not possible to take into consideration other performance measures at the same time, as, for instance, the time needed to find the optimum. EMOPaT extends classical Meta-EAs by considering more objectives at the same time. This set of objectives (represented by the block "Quality" in Figure 3) can include any of:

- different functions;
- the same function with different constraints, like time limitations;
- different quality indices used to assess the results of the optimization of a function, such as the fitness value, or the time needed to reach a preset fitness value.

Compared to classical Meta-EA approaches, the result of the process is not only a good set of parameters for solving the problem(s) under consideration, but a population of parameter sets, non-dominated according to the objectives taken into consideration. A parameter set is called non-dominated when no other solution has been found that is better on all objectives. Therefore, each individual in the NSGA-II population represents a good solution, yielding a good trade-off between conflicting goals. The actual result of NSGA-II is then the non-dominated subset of the whole population, which ideally should sample and approximate the Pareto front for the problem at hand, i.e., the whole set of all possible non-dominated solutions.

The analysis of the solutions found by the multi-objective approach to parameter tuning can also provide information on how a parameter affects an EA, how to infer novel solutions from the ones found, and how to generalize them. In [13], by looking at the correlations between parameters and fitness values, we were able to draw same interesting general conclusions about the dependence of the metaheuristics' performance on the value of some parameters, finding, for instance, that the smaller the population, the faster a good solution can be found, while the increase in the number of solutions generally leads to more precise results.

Another interesting ability of EMOPaT is that, unlike classical Meta-EAs, it is able to deal with nominal parameters, such as PSO topology or the kind of crossover used in DE. This is done by reserving in the representation of a NSGA-II individual a real number for each possible choice, and then selecting the parameter setting associated with the highest value.

In [13], EMOPaT has proven its ability in different scenarios, optimizing classical benchmark functions. Its results, besides giving useful hints for understanding parameter semantics, were sets of parameters that compared favorably

with other manually- and automatically-tuned sets of parameters. This is the first time we apply EMOPaT to a real-world problem.

5 Experimental Setup

Our goal is to find sets of PSO parameters that are able, according to the needs of the application and the hardware on which it runs, to find a good alignment between point clouds in a limited amount of time. For this reason, the two objectives we let EMOPaT optimize are:

1. Minimum time required to reach a fitness value of 0.1, averaged over 10 repetitions;
2. Best fitness reached after a time limit of 6 seconds, averaged over 10 repetitions.

Tests were run on a machine equipped with a 64-bit Intel Core i7 CPU running at 3.40 GHz using CUDA v. 5.0 on an nVidia GeForce GTX680 graphics card with 1536 cores working at 1.20 GHz and compute capability 3.0. PSO was implemented on CUDA using the open source library LibCudaOptimize [9] and running on GPU, while NSGA-II was implemented in C++, since the advantages of implementing it on GPU would not have been significant.

NSGA-II was run with these parameters: 60 individuals, 30 generations, mutation rate = 0.125, crossover rate = 0.9. The ranges within which PSO parameters were allowed to vary are shown in Table 1. Referring to Figure 3, NSGA-II is the Tuner EA, PSO is the implemented metaheuristic, and instead of a single quality index we have two of them, running time and final fitness.

Table 1. Ranges of PSO parameters delimiting the search space of the tuning algorithms. During NSGA-II evolution, they are normalized in the range $[0, 1]$ and linearly scaled in the correct range when the PSO instance is created.

Parameter	Range
Population Size	$[4, 300]$
Inertia Factor (w)	$[-0.5, 2.0]$
c_1	$[-1.0, 4.0]$
c_2	$[-1.0, 4.0]$
Topology	$\{ring, star, global\}$

The function to optimize is a simple instance of the problem presented in Section 3. A point cloud representing a wooden mallet is randomly roto-translated and used as target. The same point cloud is used as reference, so it is possible to reach a perfect alignment if PSO is able to find the correct roto-translation.

6 Results

Figure 4 shows the results obtained by EMOPaT. Each point represents the result of a set of parameters on the two objectives. The ones highlighted are the non-dominated ones that approximate the Pareto front and are the ones to be taken into consideration.

Let us have a brief look at these solutions. The main difference that we can observe between parameter sets that converge quickly to a solution with respect to the ones that reach a high final precision is that the former have a smaller population. This confirms the results we obtained on benchmark functions [13]; small populations are good at reaching quickly a good fitness value, but are generally unable to refine it because they are more likely to get stuck into local minima. Nevertheless, good populations are usually smaller than the ones usually suggested by common rules of thumb (e.g. ten times the problem dimension).

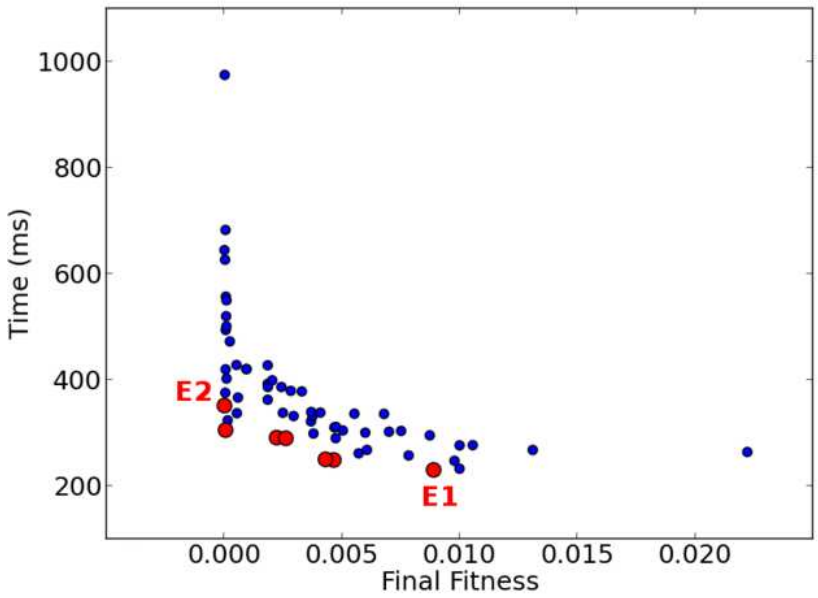

Fig. 4. Results of EMOPaT. Larger, red dots represent the solutions approximating the Pareto front. E_1 and E_2 indicate the "fastest" and the "most precise" solutions, respectively.

We selected the two solutions which lie on opposite ends of the Pareto Front approximation (indicated by E_1 and E_2 in Figure 4), and tested them more deeply on the point-cloud alignment problem. To prove the correctness of the results, we compared the parameters found by EMOPaT with the ones used in [14]. The latter parameters had been selected as the ones that performed best

within a set of 40 manually-chosen combinations. These parameters, as well as the ones found by EMOPaT are listed in Table 2.

Table 2. Manually tuned (M_1 and M_2) PSO instances and instances optimized by EMOPaT (E_1 and E_2). The termination criterion is time (0.3, 0.7, 1.3, 2.3 and 3.2 seconds).

Parameter	M_1	M_2	E_1	E_2
Population Size	24	24	15	22
Inertia Factor (w)	0.5	0.7	0.6652	0.5944
c_1	1.19	1.8	1.0479	2.1320
c_2	1.19	0.7	0.4271	0.7120
Topology	*Ring*	*Global*	*Global*	*Global*

To test them, we performed the same task used in the optimization process using each set of parameters with different time limits, similarly to what we have done in [14]. The time limits we chose were 0.3, 0.7, 1.3, 2.3 and 3.2 seconds.

Table 3 shows, for each PSO configuration and for each time limit, the average fitness and standard deviation computed over 100 independent repetitions. We performed the Wilcoxon signed-rank test ($p = 0.01$) between results at each time limit to see if there were significant differences; for each time limit, the best-performing PSOs are highlighted in bold.

Table 3. Average and standard deviation of fitness. Each column shows data of a PSO version, each row shows all data obtained at the corresponding time limit. PSO versions that perform statistically better are highlighted in bold.

PSO →	M_1		M_2		E_1		E_2	
Time ↓	Avg	Std	Avg	Std	Avg	Std	Avg	Std
0.3 s	1.62e-01	9.34e-02	1.24e-01	5.89e-02	**7.66e-02**	**9.11e-02**	1.10e-01	1.12e-01
0.7 s	4.37e-02	2.96e-02	3.65e-02	2.41e-02	**2.48e-02**	**2.64e-02**	**2.05e-02**	**1.54e-02**
1.3 s	1.80e-02	1.55e-02	1.41e-02	2.61e-02	1.68e-02	2.01e-02	**9.30e-03**	**1.62e-02**
2.3 s	**7.48e-03**	**7.83e-03**	**7.64e-03**	**1.24e-02**	1.30e-02	1.44e-02	**6.65e-03**	**1.42e-02**
3.2 s	**4.02e-03**	**6.36e-03**	**5.65e-03**	**1.04e-02**	1.29e-02	1.25e-02	**4.53e-03**	**8.62e-03**

These results prove the correctness of our multi-objective approach. The set of parameters E_1, which was the fastest to reach the fitness goal, is in fact the best-performing one after 0.3 seconds, but it is not able to further improve its results. At the end of evolution, E_2 is comparable to the two manually-selected ones. The main difference between E_2 and the latter is clear if one considers the intermediate time limits, on which E_2 performs better than M_1 and M_2. This is a clear advantage of using EMOPaT. Eventually, many parameter combinations are able to constantly reach a perfect solution if a sufficient amount of time is given. EMOPaT has the ability to find, among all these possible solutions, the ones that are also able to reach a good fitness as fast as possible, because it also optimizes the other objective(s), that are not taken into consideration in a

single-objective optimization or are very difficult to account for when performing manual tuning.

The fitness we chose is only a surrogate that reflects the real goal, which is to align the poses of the target and the reference point clouds. In Figure 5 we present the same results from a different point of view. The plots show the average error in terms of translation (euclidean distance between centers of gravity of target and reference at the end of optimization) and rotation (angle between the orientations of target and reference). These data allow one to draw the same conclusions and confirm the correctness of the proposed approach.

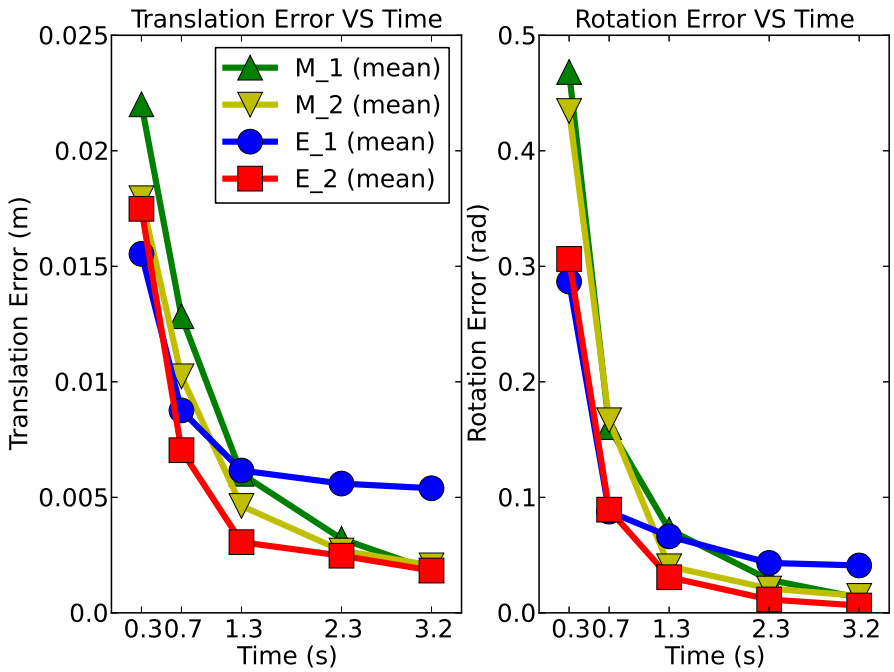

Fig. 5. Results of the four PSOs with different time budgets.

7 Final remarks

In this paper we proved the ability of multi-objective automatic parameter tuning to optimize PSO's performance on a real-world task, the localization of an object represented as a point cloud.

Using a multi-objective metaheuristic, we were able to obtain a "fast" PSO version that reached good results in a timely manner and an "optimized" PSO

which, at convergence, reached results comparable to the manually-tuned solutions, but more quickly, because in the optimization phase this ability was explicitly requested. This proves the potentiality of our multi-objective approach to parameter tuning in real-world optimization tasks, in which the best compromise between conflicting goals such as time and quality of results is usually requested.

References

1. Bartz-Beielstein, T., Lasarczyk, C., Preuss, M.: Sequential parameter optimization. In: IEEE Congress on Evolutionary Computation. vol. 1, pp. 773–780 Vol.1 (2005)
2. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A classification of hyper-heuristic approaches. In: Handbook of Metaheuristics, pp. 449–468. Springer (2010)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
4. Eiben, A.E., Smit, S.K.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm and Evolutionary Computation 1(1), 19 – 31 (2011)
5. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proc. IEEE International Conference on Neural Networks. vol. 4, pp. 1942–1948 (1995)
6. Li, H., Shen, T., Huang, X.: Approximately global optimization for robust alignment of generalized shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(6), 1116–1131 (2011)
7. López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2011-004 (2011)
8. Makadia, A., Patterson, A., Daniilidis, K.: Fully automatic registration of 3D point clouds. In: Conf. on Computer Vision and Pattern Recognition. pp. 1297–1304 (2006)
9. Nashed, Y.S.G., Ugolotti, R., Mesejo, P., Cagnoni, S.: libCudaOptimize: an open source library of GPU-based metaheuristics. In: Proceedings of the 14th international conference on Genetic and evolutionary computation conference (GECCO) companion. pp. 117–124 (2012)
10. Oleari, F., Lodi Rizzini, D., Caselli, S.: A low-cost stereo system for 3D object recognition. In: IEEE International Conference on Intelligent Computer Communication and Processing (ICCP). pp. 127–132 (2013)
11. Rusu, R.B., Blodow, N., Beetz, M.: Fast point feature histograms (FPFH) for 3D registration. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 3212–3217 (2009)
12. Storn, R., Price, K.: Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. rep., International Computer Science Institute (1995)
13. Ugolotti, R., Cagnoni, S.: Analysis of Evolutionary Algorithms using Multi-Objective Parameter Tuning. In: Proc. of the Genetic and Evolutionary Computation Conference. GECCO (2014)
14. Ugolotti, R., Micconi, G., Aleotti, J., Cagnoni, S.: GPU-based Point Cloud Recognition using Evolutionary Algorithms. In: European Conference on the Applications of Evolutionary Computation. EvoApps (2014)

15. Ugolotti, R., Nashed, Y.S.G., Mesejo, P., Ivekovič, Š., Mussi, L., Cagnoni, S.: Particle Swarm Optimization and Differential Evolution for model-based object detection. Applied Soft Computing 13(6), 3092–3105 (2013)
16. Urfalolu, O., Mikulastik, P., Stegmann, I.: Scale Invariant Robust Registration of 3D-Point Data and a Triangle Mesh by Global Optimization. In: Advanced Concepts for Intelligent Vision Systems, Lecture Notes in Computer Science, vol. 4179, pp. 1059–1070. Springer Berlin Heidelberg (2006)