合肥学院
HEFEI UNIVERSITY

Metaheuristic Optimization

17. Particle Swarm Optimization

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2	合肥学院 南艳湖校区/南2区
Faculty of Computer Science and Technology	计算机科学与技术系
Institute of Applied Optimization	应用优化研究所
230601 Shushan District, Hefei, Anhui, China	中国 安徽省 合肥市 蜀山区 230601
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99	经济技术开发区 锦绣大道99号

Outline

1. Introduction
2. Basic Phenomena
3. Particle Swarm Optimization

website

Section Outline

1. Introduction
2. Basic Phenomena
3. Particle Swarm Optimization

Introduction

- Swarms of tiny, simple creatures able to colossal achievements

Introduction

- Swarms of tiny, simple creatures able to colossal achievements
- Self-Organization

- Swarms of tiny, simple creatures able to colossal achievements
- Self-Organization
- Swarm Intelligence (SI) methods make use of these phenomena for optimization [1–4]

Section Outline

Definition (Emergence)

Emergence is the spontaneous assemblence of new properties or structures on a macro-level of a system as a result of the joint behavior of its elements on a micro-level. Emergent properties cannot be traced back to the properties that the elements of a system's micro-level exhibit in an isolated state in an obvious manner.

Emergence

Definition (Emergence)

Emergence is the spontaneous assemblence of new properties or structures on a macro-level of a system as a result of the joint behavior of its elements on a micro-level. Emergent properties cannot be traced back to the properties that the elements of a system's micro-level exhibit in an isolated state in an obvious manner.

In other words:

Emergence

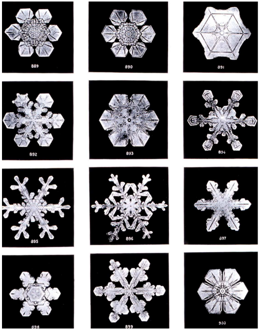
Definition (Emergence)

Emergence is the spontaneous assemblence of new properties or structures on a macro-level of a system as a result of the joint behavior of its elements on a micro-level. Emergent properties cannot be traced back to the properties that the elements of a system's micro-level exhibit in an isolated state in an obvious manner.

In other words:
The whole is more than the sum of its parts.

The whole is more than the sum of its parts.

The whole is more than the sum of its parts.

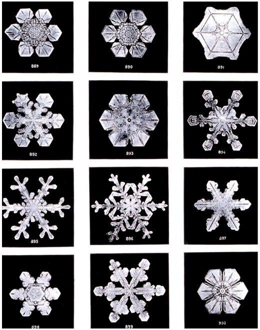


single atomes arranged due to the laws of physics form a geometric structure which is not related to the features of the single atomes in any obvious way

pictures: `http://en.wikipedia.org/wiki/Emergence`

Emergence

The whole is more than the sum of its parts.



single atomes arranged due to the laws of physics form a geometric structure which is not related to the features of the single atomes in any obvious way

single termites aggregate pieces of clay, forming a giant nest whose structure is not obviously related to the behavioral pat-terns of a single termite

pictures: `http://en.wikipedia.org/wiki/Emergence`

Definition (Self-Organization)

Self-organization is a process where constraints, shape-providing influences, and/or designing influences during the process of the creation or modification of a system come from the system itself.

Definition (Self-Organization)

Self-organization is a process where constraints, shape-providing influences, and/or designing influences during the process of the creation or modification of a system come from the system itself.

Definition (Self-Organization)

Self-organization is a process where constraints, shape-providing influences, and/or designing influences during the process of the creation or modification of a system come from the system itself.

A swarm results from the interactions of the single birds without the need of any "lead bird" or controler outside of the swarm

Section Outline

- Wilson [5] states about fish schools: *"In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantages of competition for food items, whenever the resource is unpredictably distributed in patches."* [6]
  (school = swarm of fish)

Swarms and PSO

- Wilson [5] states about fish schools: *"In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantages of competition for food items, whenever the resource is unpredictably distributed in patches."* [6]
  (school = swarm of fish)
- Particle Swarm Optimization (PSO) [6–14] was developed by Eberhart and Kennedy [6, 10, 11] in 1995 to make use of this phenomenon for optimization

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems

Particle Swarm Optimization

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems
- Search space is subset of real vectors numbers: $\mathbb{G} \subseteq \mathbb{R}^n$

Particle Swarm Optimization

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems
- Search space is subset of real vectors numbers: $\mathbb{G} \subseteq \mathbb{R}^n$
- Population pop = swarm of particles p which move in $\mathbb{G}$

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems
- Search space is subset of real vectors numbers: $\mathbb{G} \subseteq \mathbb{R}^n$
- Population pop = swarm of particles p which move in $\mathbb{G}$
- Genotype $p.g$ = position of particle p

Particle Swarm Optimization

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems
- Search space is subset of real vectors numbers: $\mathbb{G} \subseteq \mathbb{R}^n$
- Population pop = swarm of particles p which move in $\mathbb{G}$
- Genotype $p.g$ = position of particle p
- Resulting Algorithm is a little bit similar to the Evolution Strategy with endogeneous information:

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems
- Search space is subset of real vectors numbers: $\mathbb{G} \subseteq \mathbb{R}^n$
- Population pop = swarm of particles p which move in $\mathbb{G}$
- Genotype $p.g$ = position of particle p
- Resulting Algorithm is a little bit similar to the Evolution Strategy with endogeneous information:
- Information about particles p and population pop:

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems
- Search space is subset of real vectors numbers: $\mathbb{G} \subseteq \mathbb{R}^n$
- Population pop = swarm of particles p which move in $\mathbb{G}$
- Genotype $p.g$ = position of particle p
- Resulting Algorithm is a little bit similar to the Evolution Strategy with endogeneous information:
- Information about particles p and population pop::
  - Endogenous information: velocity vectors of the particles $p.\vec{v}$

Particle Swarm Optimization

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems
- Search space is subset of real vectors numbers: $\mathbb{G} \subseteq \mathbb{R}^n$
- Population pop = swarm of particles p which move in $\mathbb{G}$
- Genotype $p.g$ = position of particle p
- Resulting Algorithm is a little bit similar to the Evolution Strategy with endogeneous information:
- Information about particles p and population pop::
  - Endogenous information: velocity vectors of the particles $p.\vec{v}$
  - $\mathrm{N}(p)$: neighbors of particle p

Particle Swarm Optimization

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems
- Search space is subset of real vectors numbers: $\mathbb{G} \subseteq \mathbb{R}^n$
- Population pop = swarm of particles p which move in $\mathbb{G}$
- Genotype $p.g$ = position of particle p
- Resulting Algorithm is a little bit similar to the Evolution Strategy with endogeneous information:
- Information about particles p and population pop::
  - Endogenous information: velocity vectors of the particles $p.\vec{v}$
  - $\mathrm{N}(p)$: neighbors of particle p
  - $\mathrm{best}(p)$: the particle's best ever position

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems
- Search space is subset of real vectors numbers: $\mathbb{G} \subseteq \mathbb{R}^n$
- Population pop = swarm of particles p which move in $\mathbb{G}$
- Genotype $p.g$ = position of particle p
- Resulting Algorithm is a little bit similar to the Evolution Strategy with endogeneous information:
- Information about particles p and population pop::
  - Endogenous information: velocity vectors of the particles $p.\vec{v}$
  - $\mathrm{N}(p)$: neighbors of particle p
  - $\mathrm{best}(p)$: the particle's best ever position
  - $\mathrm{best}(\mathrm{pop})$: the best position ever found in the population

Particle Swarm Optimization

- Copy the behavior with which swarms / schools / flocks in nature find food for solving optimization problems
- Search space is subset of real vectors numbers: $\mathbb{G} \subseteq \mathbb{R}^n$
- Population $\mathrm{pop} =$ swarm of particles p which move in $\mathbb{G}$
- Genotype $p.g =$ position of particle p
- Resulting Algorithm is a little bit similar to the Evolution Strategy with endogeneous information:
- Information about particles p and population pop::
  - Endogenous information: velocity vectors of the particles $p.\vec{v}$
  - $\mathrm{N}(p)$: neighbors of particle p
  - $\mathrm{best}(p)$: the particle's best ever position
  - $\mathrm{best}(\mathrm{pop})$: the best position ever found in the population
  - $\mathrm{best}(\mathrm{N}(p))$: the best position ever found in the neighborhood of p

Velocity Update

- When updating a particle p, first it's velocity $p.\vec{v}$ is updated

Velocity Update

- When updating a particle p, first it's velocity $p.\vec{v}$ is updated
- There are different two ways to do that

Velocity Update

- When updating a particle p, first it's velocity $p.\vec{v}$ is updated
- There are different two ways to do that:
  1. Local Update: based on the particle's current velocity, it's best position in history, and the best position of its neighbors

- When updating a particle p, first it's velocity $p.\vec{v}$ is updated
- There are different two ways to do that:
  1. Local Update: based on the particle's current velocity, it's best position in history, and the best position of its neighbors:

$$\begin{aligned} p.\vec{v}_i = p.\vec{v}_i + \; & [\{\text{randomly from } [0, c]\} * (\mathrm{best}(p).g_i - p.g_i)] + \\ & [\{\text{randomly from } [0, d]\} * (\mathrm{best}(\mathrm{N}(p)).g_i - p.g_i)] \end{aligned} \quad (1)$$

- When updating a particle p, first it's velocity $p.\vec{v}$ is updated
- There are different two ways to do that:
  1. Local Update: based on the particle's current velocity, it's best position in history, and the best position of its neighbors:

$$p.\vec{v}_i = p.\vec{v}_i + \begin{array}{l} [\{\text{randomly from } [0,c]\} * (\text{best}(p).g_i - p.g_i)] + \\ [\{\text{randomly from } [0,d]\} * (\text{best}(\text{N}(p)).g_i - p.g_i)] \end{array} \quad (1)$$

  2. Global Update: based on the best position in the population

Velocity Update

- When updating a particle p, first it's velocity $p.\vec{v}$ is updated
- There are different two ways to do that:
  1. Local Update: based on the particle's current velocity, it's best position in history, and the best position of its neighbors:

$$p.\vec{v}_i = p.\vec{v}_i + \begin{array}{l} [\{\text{randomly from } [0, c]\} * (\text{best}(p).g_i - p.g_i)] + \\ [\{\text{randomly from } [0, d]\} * (\text{best}(\text{N}(p)).g_i - p.g_i)] \end{array} \quad (1)$$

  2. Global Update: based on the best position in the population:

$$p.\vec{v}_i = p.\vec{v}_i + \begin{array}{l} [\{\text{randomly from } [0, c]\} * (\text{best}(p).g_i - p.g_i)] + \\ [\{\text{randomly from } [0, d]\} * (\text{best}(\text{pop}).g_i - p.g_i)] \end{array} \quad (2)$$

Velocity Update

- When updating a particle p, first it's velocity $p.\vec{v}$ is updated
- There are different two ways to do that:
  1. Local Update: based on the particle's current velocity, it's best position in history, and the best position of its neighbors:

$$p.\vec{v}_i = p.\vec{v}_i + \begin{array}{l}[\{\text{randomly from } [0,c]\} * (\text{best}(p).g_i - p.g_i)] + \\ [\{\text{randomly from } [0,d]\} * (\mathbf{best(N(}\boldsymbol{p}\mathbf{)).}\boldsymbol{g_i} - p.g_i)]\end{array} \quad (1)$$

  2. Global Update: based on the best position in the population:

$$p.\vec{v}_i = p.\vec{v}_i + \begin{array}{l}[\{\text{randomly from } [0,c]\} * (\text{best}(p).g_i - p.g_i)] + \\ [\{\text{randomly from } [0,d]\} * (\mathbf{best(pop).}\boldsymbol{g_i} - p.g_i)]\end{array} \quad (2)$$

- Social Component: information exchange with other particles

- When updating a particle p, first it's velocity $p.\vec{v}$ is updated
- There are different two ways to do that:
  1. Local Update: based on the particle's current velocity, it's best position in history, and the best position of its neighbors:

$$p.\vec{v}_i = p.\vec{v}_i + \begin{array}{l}[\{\text{randomly from } [0,c]\} * (\mathrm{best}(p).g_i - p.g_i)] + \\ [\{\text{randomly from } [0,d]\} * (\mathbf{best(N(}\boldsymbol{p}\mathbf{)).}\boldsymbol{g_i} - p.g_i)]\end{array} \quad (1)$$

  2. Global Update: based on the best position in the population:

$$p.\vec{v}_i = p.\vec{v}_i + \begin{array}{l}[\{\text{randomly from } [0,c]\} * (\mathrm{best}(p).g_i - p.g_i)] + \\ [\{\text{randomly from } [0,d]\} * (\mathbf{best(pop).}\boldsymbol{g_i} - p.g_i)]\end{array} \quad (2)$$

- Social Component: information exchange with other particles
- Parameters $c, d \in [0, 1]$ influence convergence speed

Velocity Update

- When updating a particle p, first it's velocity $p.\vec{v}$ is updated
- There are different two ways to do that:
  1. Local Update: based on the particle's current velocity, it's best position in history, and the best position of its neighbors:

$$p.\vec{v}_i = p.\vec{v}_i + \begin{array}{l}[\{\text{randomly from } [0,c]\} * (\mathrm{best}(p).g_i - p.g_i)] + \\ [\{\text{randomly from } [0,d]\} * (\mathbf{best(N(}\boldsymbol{p}\mathbf{)).}\boldsymbol{g_i} - p.g_i)]\end{array} \quad (1)$$

  2. Global Update: based on the best position in the population:

$$p.\vec{v}_i = p.\vec{v}_i + \begin{array}{l}[\{\text{randomly from } [0,c]\} * (\mathrm{best}(p).g_i - p.g_i)] + \\ [\{\text{randomly from } [0,d]\} * (\mathbf{best(pop).}\boldsymbol{g_i} - p.g_i)]\end{array} \quad (2)$$

- Social Component: information exchange with other particles
- Parameters $c, d \in [0, 1]$ influence convergence speed
- Warning: Velocity may increase without bound ... update must bound velocity into a $[min, max]$ interval!

PSO Algorithm

- The second step of updating a particle is to update its position (genotype)

- The second step of updating a particle is to update its position (genotype):

$$p.g_i = p.g_i + p.\vec{v}_i \tag{3}$$

PSO Algorithm

- The second step of updating a particle is to update its position (genotype):

$$p.g_i = p.g_i + p.\vec{v}_i \tag{3}$$

- The PSO algorithm works as follows

- The second step of updating a particle is to update its position (genotype):

$$p.g_i = p.g_i + p.\vec{v}_i \tag{3}$$

- The PSO algorithm works as follows

$p_{best} \longleftarrow \mathrm{PSO}(f, ps)$

begin
- pop $\longleftarrow$ create population of ps particles
- **while** $\neg shouldTerminate$ **do**
  - **for** $i \longleftarrow 0$ **up to** $ps - 1$ **do**
    - $\mathrm{pop}[i] \longleftarrow \mathrm{psoUpdate}(\mathrm{pop}[i], \mathrm{pop})$
- **return** $best(pop)$

PSO Individual Record

Listing: The PSO Individual Record

```
public class PSOIndividual<X> extends Individual<double[], X> {

  /** the velocity vector */
  public final double[] velocity;

  /** the best position seen by this individual */
  public final Individual<double[], X> best;
}
```

Listing: The PSO Algorithm

```java
public class PSO<X> extends OptimizationAlgorithm<double[], X> {
  public Individual<double[], X> solve(final IObjectiveFunction<X> f) {
    final PSOIndividual<X>[] swarm;
    PSOIndividual<X> cur;
    Individual<double[], X> best;
    double limitV;
    int i, j;

    swarm = new PSOIndividual[this.ps];
    best = new Individual<>();
    best.g = new double[this.rn.dim];

    limitV = 0.1 * (this.rn.max - this.rn.min);

    for (i = swarm.length; (--i) >= 0;) {
      swarm[i] = cur = new PSOIndividual<>(this.nullary.create(this.random));
      cur.x = this.gpm.gpm(cur.g);
      cur.v = f.compute(cur.x);
      copyIndividual(cur.best, cur);
      if (cur.v < best.v) {
        copyIndividual(best, cur);
      }

      if (this.termination.shouldTerminate()) {
        return best;
      }
    }

    for (;;) {

      for (i = swarm.length; (--i) >= 0;) {
        cur = swarm[i];
        for (j = this.rn.dim; (--j) >= 0;) {
          cur.velocity[j] = Math.min(limitV, Math.max(-limitV,
              cur.velocity[j] = ((this.random.nextDouble() * this.c) * (cur.best.g[j] - cur.g[j]))
                  + ((this.random.nextDouble() * this.d) * (best.g[j] - cur.g[j]))));
        }
      }

      for (i = swarm.length; (--i) >= 0;) {
        cur = swarm[i];
        for (j = this.rn.dim; (--j) >= 0;) {
          cur.g[j] = Math.max(this.rn.min, Math.min(this.rn.max, cur.g[j] + cur.velocity[j]));
          cur.x = this.gpm.gpm(cur.g);
          cur.v = f.compute(cur.x);
          if (cur.v < cur.best.v) {
            copyIndividual(cur.best, cur);
            if (cur.v < best.v) {
              copyIndividual(best, cur);
            }
          }
          if (this.termination.shouldTerminate()) {
            return best;
          }
        }
      }
    }
  }
```

- PSO is a simple numerical optimization algorithm

Summary

- PSO is a simple numerical optimization algorithm
- But: PSO is *not* rotationally invariant! [15]

Summary

- PSO is a simple numerical optimization algorithm
- But: PSO is *not* rotationally invariant! [15]
    - It performs well on (axis-parallel) separable functions (potentially better than CMA-ES)

Summary

- PSO is a simple numerical optimization algorithm
- But: PSO is *not* rotationally invariant! [15]
  - It performs well on (axis-parallel) separable functions (potentially better than CMA-ES)
  - But much worse if the same functions are rotated or the problems are non-separable (epistatic) [15]

谢谢
Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. Christian Blum and Daniel Merkle, editors. *Swarm Intelligence – Introduction and Applications*. Natural Computing Series. New York, NY, USA: Springer New York, 2008. ISBN 978-3-540-74088-9 and 978-3-540-74089-6. doi: 10.1007/978-3-540-74089-6. URL http://books.google.de/books?id=6Ky4bVPCXqMC.
2. Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. New York, NY, USA: John Wiley & Sons Ltd., 2005. ISBN 0470091916 and 978-0470091913. URL http://books.google.de/books?id=UAg_AAAACAAJ.
3. James Kennedy and Russel C. Eberhart. *Swarm Intelligence: Collective, Adaptive*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. ISBN 1558605959 and 9781558605954. URL http://www.engr.iupui.edu/~eberhart/web/PSObook.html.
4. Eric W. Bonabeau, Marco Dorigo, and Guy Théraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. New York, NY, USA: Oxford University Press, Inc., August 1999. ISBN 0195131592 and 9780195131598. URL http://books.google.de/books?id=mhLj70l8HfAC.
5. Edward Osborne Wilson. *Sociobiology: The New Synthesis*. Cambridge, MA, USA: Belknap Press and Cambridge, MA, USA: Harvard University Press, 1975. ISBN 0674000897 and 9780674000896. URL http://books.google.de/books?id=v7lV9tz8fXAC.
6. James Kennedy and Russel C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN'95)*, volume 4, pages 1942–1948, Perth, WA, Australia: University of Western Australia, November 27–December 1, 1995. Los Alamitos, CA, USA: IEEE Computer Society Press. doi: 10.1109/ICNN.1995.488968. URL http://www.engr.iupui.edu/~shi/Coference/psopap4.html.
7. Gerhard Venter and Jaroslaw Sobieszczanski-Sobieski. Particle swarm optimization. *AIAA Journal*, 41(8):1583–1589, August 2003. URL http://pdf.aiaa.org/getfile.cfm?urlX=2%3CWIG7D%2FQKU%3E6B5%3AKF5%2B%5CQ%3AK%3E%0A&urla=%26%2B%22L%2F%22P%22%40%0A&urlb=%21%2A%20%20%20%0A&urlc=%21%2A0%20%20%0A&urld=%21%2A0%20%20%0A&urle=%27%28%22H%22%23PJAT%20%20%20%0A.
8. Tao Cai, Feng Pan, and Jie Chen. Adaptive particle swarm optimization algorithm. In *Proceedings of Fifth World Congress on Intelligent Control and Automation (WCICA'04)*, volume 3, pages 2245–2247, Hangzhou, Zhejiang, China, June 15–19, 2004. Piscataway, NJ, USA: IEEE (Institute of Electrical and Electronics Engineers). doi: 10.1109/WCICA.2004.1341988.

Bibliography II

9. Yuelin Gao and Yuhong Duan. An adaptive particle swarm optimization algorithm with new random inertia weight. In De-Shuang Huang, Laurent Heutte, and Marco Loog, editors, *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques – Proceedings of the Third International Conference on Intelligent Computing (ICIC'07-2)*, volume 2 of *Communications in Computer and Information Science*, pages 342–350, Qingdao, Shandong, China, August 21–24, 2007. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-540-74282-1_39.
10. Russel C. Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS'95)*, pages 39–43, Nagoya, Aichi, Japan, October 4–6, 1995. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/MHS.1995.494215. URL `http://webmining.spd.louisville.edu/Websites/COMB-OPT/FINAL-PAPERS/SwarmsPaper.pdf`.
11. Russel C. Eberhart and Yuhui Shi. A modified particle swarm optimizer. In Patrick K. Simpson, editor, *The 1998 IEEE International Conference on Evolutionary Computation (CEC'98), 1998 IEEE World Congress on Computation Intelligence (WCCI'98)*, pages 69–73, Anchorage, AK, USA: Egan Civic & Convention Center and Anchorage Hilton Hotel, May 4–9, 1998. Piscataway, NJ, USA: IEEE Computer Society, Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/ICEC.1998.699146.
12. Yuelin Gao and Zihui Ren. Adaptive particle swarm optimization algorithm with genetic mutation operation. In Jingsheng Lei, JingTao Yao, and Qingfu Zhang, editors, *Proceedings of the Third International Conference on Advances in Natural Computation (ICNC'07)*, volume 2, pages 211–215, Haikou, Hainan, China, August 24–27, 2007. Los Alamitos, CA, USA: IEEE Computer Society Press. doi: 10.1109/ICNC.2007.161.
13. Russel C. Eberhart and Yuhui Shi. Comparison between genetic algorithms and particle swarm optimization. In Vincent William Porto, N. Saravanan, D. Waagen, and Ágoston E. Eiben, editors, *Evolutionary Programming VII – Proceedings of the 7th International Conference on Evolutionary Programming (EP'98)*, volume 1447/1998 of *Lecture Notes in Computer Science (LNCS)*, pages 611–616, San Diego, CA, USA: Mission Valley Marriott, May 25–27, 1998. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/BFb0040812.
14. Peter John Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In Vincent William Porto, N. Saravanan, D. Waagen, and Ágoston E. Eiben, editors, *Evolutionary Programming VII – Proceedings of the 7th International Conference on Evolutionary Programming (EP'98)*, volume 1447/1998 of *Lecture Notes in Computer Science (LNCS)*, pages 601–610, San Diego, CA, USA: Mission Valley Marriott, May 25–27, 1998. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/BFb0040811.

Bibliography III

15. Nikolaus Hansen, Raymond Ros, Nikolas Mauny, Marc Schoenauer, and Anne Auger. Impacts of invariance in search: When cma-es and pso face ill-conditioned and non-separable problems. 11(8):5755–5769, December 2011. doi: 10.1016/j.asoc.2011.03.001. URL hal.inria.fr/inria-00583669/PDF/hansen2011impacts.pdf. INRIA Report inria-00583669, version 1, 2011-04-06.