Universidade do Minho

Particle swarm algorithms for multi-local optimization

A. Ismael F. Vaz

Edite M.G.P. Fernandes

Production and System Department

Engineering School

Minho University

{aivaz,emgpf}@dps.uminho.pt

Work partially supported by FCT grant POCI/MAT/58957/2004 and Algoritmi research center

Universidade do Minho

Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions

Universidade do Minho

Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions

Universidade do Minho

Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions

Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions

Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions

Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions

Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions

Motivation

One of the (many) applications of multi-local optimization is in reduction type methods for semi-infinite programming (SIP) problems.

Motivation

One of the (many) applications of multi-local optimization is in reduction type methods for semi-infinite programming (SIP) problems.

A SIP problems can be posed as:

$$\min_{y \in R^q} \; o(y)$$
$$s.t. \;\; f_i(y,x) \geq 0, \;\; i = 1, \ldots, m$$
$$\forall x \in T \subset R^n,$$

where $o(y)$ is the objective function and $f_i(y,x)$, $i = 1, \ldots, m$, are the infinite constraint functions.

Universidade do Minho

Motivation

One of the (many) applications of multi-local optimization is in reduction type methods for semi-infinite programming (SIP) problems.

A SIP problems can be posed as:

$$\min_{y \in R^q} \; o(y)$$
$$s.t. \;\; f_i(y,x) \geq 0, \; i = 1, \ldots, m$$
$$\forall x \in T \subset R^n,$$

where $o(y)$ is the objective function and $f_i(y,x)$, $i = 1, \ldots, m$, are the infinite constraint functions.

A feasible point must satisfy:

$$f_i(y,x) \geq 0, \; i = 1, \ldots, m, \; \forall x \in T$$

meaning that the global minima of f_i must be upper than or equal to zero.

Multi-local optimization

Assume, for the sake of simplicity, that $m = 1$. Then we want to address the following optimization problem

$$\min_{x \in R^n} f(x)$$
$$s.t. \;\; a \leq x \leq b$$

where $f : R^n \rightarrow R$ is the objective function and a, b are the simple bounds on the variables x (defining the set T).

Multi-local optimization

Assume, for the sake of simplicity, that $m = 1$. Then we want to address the following optimization problem

$$\min_{x \in R^n} f(x)$$
$$s.t. \;\; a \le x \le b$$

where $f : R^n \to R$ is the objective function and a, b are the simple bounds on the variables x (defining the set T).

In each iteration of a reduction type method for SIP we need to obtain all the feasible global and local optima for function $f(x)$.

The Particle Swarm Paradigm (PSP)

The PSP is a population (swarm) based algorithm that mimics the social behavior of a set of individuals (particles).

An individual behavior is a combination of its past experience (cognition influence) and the society experience (social influence).

In the optimization context a particle p, at time instant t, is represented by its current position ($x^p(t)$), its best ever position ($y^p(t)$) and its travelling velocity ($v^p(t)$).

The new travel position and velocity

The new particle position is updated by

$$x^p(t+1) = x^p(t) + v^p(t+1),$$

where $v^p(t+1)$ is the new velocity given by

$$v_j^p(t+1) = \iota(t) v_j^p(t) + \mu\omega_{1j}(t)\left(y_j^p(t) - x_j^p(t)\right) + \nu\omega_{2j}(t)\left(\hat{y}_j(t) - x_j^p(t)\right),$$

for $j = 1, \ldots, n$.

- $\iota(t)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν is the *social* parameter
- $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are random numbers drawn from the uniform $(0, 1)$ distribution.

The new travel position and velocity

The new particle position is updated by

$$x^p(t+1) = x^p(t) + v^p(t+1),$$

where $v^p(t+1)$ is the new velocity given by

$$v_j^p(t+1) = \iota(t)v_j^p(t)+\mu\omega_{1j}(t)\left(y_j^p(t) - x_j^p(t)\right)+\nu\omega_{2j}(t)\left(\hat{y}_j(t) - x_j^p(t)\right),$$

for $j = 1, \ldots, n$.

- $\iota(t)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν is the *social* parameter
- $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are random numbers drawn from the uniform $(0, 1)$ distribution.

The new travel position and velocity

The new particle position is updated by

$$x^p(t+1) = x^p(t) + v^p(t+1),$$

where $v^p(t+1)$ is the new velocity given by

$$v_j^p(t+1) = \iota(t)v_j^p(t) + \mu\omega_{1j}(t)\left(y_j^p(t) - x_j^p(t)\right) + \nu\omega_{2j}(t)\left(\hat{y}_j(t) - x_j^p(t)\right),$$

for $j = 1, \ldots, n$.

- $\iota(t)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν is the *social* parameter
- $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are random numbers drawn from the uniform $(0, 1)$ distribution.

The new travel position and velocity

The new particle position is updated by

$$x^p(t+1) = x^p(t) + v^p(t+1),$$

where $v^p(t+1)$ is the new velocity given by

$$v_j^p(t+1) = \iota(t)v_j^p(t) + \mu\omega_{1j}(t)\left(y_j^p(t) - x_j^p(t)\right) + \nu\omega_{2j}(t)\left(\hat{y}_j(t) - x_j^p(t)\right),$$

for $j = 1, \ldots, n$.

- $\iota(t)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν is the *social* parameter
- $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are random numbers drawn from the uniform $(0,1)$ distribution.

Universidade do Minho

The best ever particle

$\hat{y}(t)$ is a particle position with global best function value so far, *i.e.*,

$$\hat{y}(t) = \arg\min_{a \in \mathcal{A}} f(a)$$
$$\mathcal{A} = \left\{ y^1(t), \ldots, y^s(t) \right\}.$$

where s is the number of particles in the swarm.

Universidade do Minho

The best ever particle

$\hat{y}(t)$ is a particle position with global best function value so far, *i.e.*,

$$\hat{y}(t) = \arg\min_{a \in \mathcal{A}} f(a)$$

$$\mathcal{A} = \left\{ y^1(t), \ldots, y^s(t) \right\}.$$

where s is the number of particles in the swarm.

In an algorithmic point of view we just have to keep track of the particle with the best ever function value.

Features

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.
2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.

Universidade do Minho

Features

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.
2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.

Universidade do Minho

Features

Population based algorithm.

1. Good
   - (a) Easy to implement.
   - (b) Easy to parallelize.
   - (c) Easy to handle discrete variables.
   - (d) Only uses objective function evaluations.
2. Not so good
   - (a) Slow rate of convergence near an optimum.
   - (b) Quite large number of function evaluations.
   - (c) In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.
2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.
2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.

Features

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.
2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.

Universidade do Minho

Features

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.
2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.

PSP with the steepest descent direction

The new particle position is updated by

$$x^p(t+1) = x^p(t) + v^p(t+1),$$

where $v^p(t+1)$ is the new velocity given by

$$v_j^p(t+1) = \iota(t)v_j^p(t) + \mu\omega_{1j}(t)\left(y_j^p(t) - x_j^p(t)\right) + \nu\omega_{2j}(t)\left(-\nabla_j f(y_j^p(t))\right),$$

for $j = 1, \ldots, n$, where $\nabla f(x)$ is the gradient of the objective function.

Each particle uses the steepest descent direction computed at each particle best position ($y^p(t)$).

The inclusion of the steepest descent direction in the velocity equation aims to drive each particle to a neighbor local minimum and since we have a population of particles, each one will be driven to a local minimum.

PSP with a descent direction

Other approach is to use

$$w^p = \frac{-1}{\sum_{k=1}^{m} |f(z_k^p) - f(y^p)|} \sum_{k=1}^{m} (f(z_k^p) - f(y^p)) \frac{(z_k^p - y^p)}{\|z_k^p - y^p\|}$$

as a descent direction at y^p, in the velocity equation, to overcome the need to compute the gradient.

Where

- y^p is the best position of particle p
- $\{z_k^p\}_{k=1}^m$ is a set of m (random) points close to y^p,

PSP with a descent direction

Other approach is to use

$$w^p = \frac{-1}{\sum_{k=1}^{m} |f(z_k^p) - f(y^p)|} \sum_{k=1}^{m} (f(z_k^p) - f(y^p)) \frac{(z_k^p - y^p)}{\|z_k^p - y^p\|}$$

as a descent direction at y^p, in the velocity equation, to overcome the need to compute the gradient.

Where

- y^p is the best position of particle p
- $\{z_k^p\}_{k=1}^m$ is a set of m (random) points close to y^p,

PSP with a descent direction

Universidade do Minho

Other approach is to use

$$w^p = \frac{-1}{\sum_{k=1}^{m} |f(z_k^p) - f(y^p)|} \sum_{k=1}^{m} (f(z_k^p) - f(y^p)) \frac{(z_k^p - y^p)}{\|z_k^p - y^p\|}$$

as a descent direction at y^p, in the velocity equation, to overcome the need to compute the gradient.

Where

- y^p is the best position of particle p
- $\{z_k^p\}_{k=1}^m$ is a set of m (random) points close to y^p,

PSP with a descent direction

Universidade do Minho

Other approach is to use

$$w^p = \frac{-1}{\sum_{k=1}^{m} |f(z_k^p) - f(y^p)|} \sum_{k=1}^{m} (f(z_k^p) - f(y^p)) \frac{(z_k^p - y^p)}{\|z_k^p - y^p\|}$$

as a descent direction at y^p, in the velocity equation, to overcome the need to compute the gradient.

Where

- y^p is the best position of particle p
- $\{z_k^p\}_{k=1}^m$ is a set of m (random) points close to y^p,

Under certain conditions w^p simulates the steepest descent direction.

Stopping criterion

We propose the stopping criterion

$$\max_p [v^p(t)]_{opt} \leq \epsilon_p$$

where

$$[v^p(t)]_{opt} = \left(\sum_{j=1}^{n} \begin{cases} 0 & \text{if } x_j^p(t) = b_j \text{ and } v_j^p(t) \geq 0 \\ 0 & \text{if } x_j^p(t) = a_j \text{ and } v_j^p(t) \leq 0 \\ \left(v_j^p(t)\right)^2 & \text{otherwise} \end{cases} \right)^{1/2}$$

Stopping criterion

Universidade do Minho

We propose the stopping criterion

$$\max_p [v^p(t)]_{opt} \leq \epsilon_p$$

where

$$[v^p(t)]_{opt} = \left(\sum_{j=1}^{n} \begin{cases} 0 & \text{if } x_j^p(t) = b_j \text{ and } v_j^p(t) \geq 0 \\ 0 & \text{if } x_j^p(t) = a_j \text{ and } v_j^p(t) \leq 0 \\ \left(v_j^p(t)\right)^2 & \text{otherwise} \end{cases} \right)^{1/2}$$

The stopping criterion is based on the optimality conditions for the multi-local optimization problem.

Environment

- Implemented in the C programming language
- Interfaced with AMPL (`www.ampl.com`)
- Both methods soon available in the NEOS server

Environment

- Implemented in the C programming language
- Interfaced with AMPL (`www.ampl.com`)
- Both methods soon available in the NEOS server

Environment

- Implemented in the C programming language
- Interfaced with AMPL (`www.ampl.com`)
- Both methods soon available in the NEOS server

Test problems

Universidade do Minho

	Problems	n	N_{x^*}	f^*		Problems	n	N_{x^*}	f^*
1	b2	2	1	0.000E+00	17	rosenbrock5	5	1	0.000E+00
2	bohachevsky	2	1	0.000E+00	18	shekel10	4	1	-1.054E+01
3	branin	2	3	3.979E-01	19	shekel5	4	1	-1.015E+01
4	dejoung	3	1	0.000E+00	20	shekel7	4	1	-1.040E+01
5	easom	2	1	-1.000E+00	21	shubert	2	18	-1.867E+02
6	f1	30	1	-1.257E+04	22	storn1	2	2	-4.075E-01
7	goldprice	2	1	3.000E+00	23	storn2	2	2	-1.806E+01
8	griewank	6	1	0.000E+00	24	storn3	2	2	-2.278E+02
9	hartmann3	3	1	-3.863E+00	25	storn4	2	2	-2.429E+03
10	hartmann6	6	1	-3.322E+00	26	storn5	2	2	-2.478E+04
11	hump	2	2	0.000E+00	27	storn6	2	2	-2.493E+05
12	hump_camel	2	2	-1.032E+00	28	zakharov10	10	1	0.000E+00
13	levy3	2	18	-1.765E+02	29	zakharov2	2	1	0.000E+00
14	parsopoulos	2	12	0.000E+00	30	zakharov20	20	1	0.000E+00
15	rosenbrock10	10	1	0.000E+00	31	zakharov4	4	1	0.000E+00
16	rosenbrock2	2	1	0.000E+00	32	zakharov5	5	1	0.000E+00

Parameters

- For each problem, the optimizer was run 5 times with different initial particle positions and velocities (randomly chosen from the search domain)
- The algorithm terminates if the stopping criterion is met with $\epsilon_p = 0.01$ or the number of iterations exceeds $N_t^{max} = 100000$
- Coefficients μ and ν were both set to 1.2
- The inertial parameter $\iota(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of N_t^{max} iterations
- The swarm size is given by $\min(6^n, 100)$, where n is the problem dimension.

Parameters

- For each problem, the optimizer was run 5 times with different initial particle positions and velocities (randomly chosen from the search domain)
- The algorithm terminates if the stopping criterion is met with $\epsilon_p = 0.01$ or the number of iterations exceeds $N_t^{max} = 100000$
- Coefficients μ and ν were both set to 1.2
- The inertial parameter $\iota(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of N_t^{max} iterations
- The swarm size is given by $\min(6^n, 100)$, where n is the problem dimension.

Parameters

Universidade do Minho

- For each problem, the optimizer was run 5 times with different initial particle positions and velocities (randomly chosen from the search domain)
- The algorithm terminates if the stopping criterion is met with $\epsilon_p = 0.01$ or the number of iterations exceeds $N_t^{max} = 100000$
- Coefficients μ and ν were both set to 1.2
- The inertial parameter $\iota(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of N_t^{max} iterations
- The swarm size is given by $\min(6^n, 100)$, where n is the problem dimension.

Parameters

- For each problem, the optimizer was run 5 times with different initial particle positions and velocities (randomly chosen from the search domain)
- The algorithm terminates if the stopping criterion is met with $\epsilon_p = 0.01$ or the number of iterations exceeds $N_t^{max} = 100000$
- Coefficients μ and ν were both set to 1.2
- The inertial parameter $\iota(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of N_t^{max} iterations
- The swarm size is given by $\min(6^n, 100)$, where n is the problem dimension.

Universidade do Minho

Parameters

- For each problem, the optimizer was run 5 times with different initial particle positions and velocities (randomly chosen from the search domain)
- The algorithm terminates if the stopping criterion is met with $\epsilon_p = 0.01$ or the number of iterations exceeds $N_t^{max} = 100000$
- Coefficients μ and ν were both set to 1.2
- The inertial parameter $\iota(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of N_t^{max} iterations
- The swarm size is given by $\min(6^n, 100)$, where n is the problem dimension.

Numerical results

Universidade do Minho

	Gradient version					Approximate descent direction version			
	$F.O.$	N_{afe}	N_{age}	f_a^*	f_{best}	$F.O.$	N_{afe}	f_a^*	f_{best}
1	100	3444343	873	0,000E+00	0,000E+00	100	3602386	0,000E+00	0,000E+00
2	100	2782058	545	0,000E+00	0,000E+00	100	3600983	0,000E+00	0,000E+00
3	100	1740823	1397	3,979E-01	3,979E-01	100	3601171	3,979E-01	3,979E-01
4	100	1647820	4420	2,618E-23	0,000E+00	100	10003223	0,000E+00	0,000E+00
5	100	283500	70615	-1,000E+00	-1,000E+00	100	3601354	-1,000E+00	-1,000E+00
6	Not differentiable					100	10104250	-1,448E+04	-1,468E+04
7	20	3600000	59	2,431E+01	4,583E+00	100	3600967	3,000E+00	3,000E+00
8	20	10000000	7754	1,084E-02	0,000E+00	0	10004487	2,257E-02	1,503E-02
9	100	10000000	483	-3,850E+00	-3,861E+00	100	10002098	-3,862E+00	-3,863E+00
10	40	10000000	525	-2,937E+00	-3,185E+00	100	10002652	-3,202E+00	-3,242E+00
11	100	963259	1082	-1,032E+00	-1,032E+00	100	3600946	-1,032E+00	-1,032E+00
12	100	1171181	1329	4,651E-08	4,651E-08	100	3601098	2,362E-06	6,756E-07
13	0	3600000	439	-1,276E+02	-1,592E+02	49	3601052	-1,765E+02	-1,765E+02
14	85	2952979	2295	4,922E-23	3,749E-33	75	3600819	2,607E-07	9,685E-08
15	0	10000000	154	8,051E+04	3,387E+04	0	10009292	8,726E+00	7,386E+00
16	0	3600000	91	3,046E+00	1,190E+00	100	3601268	1,437E-06	5,698E-07

Numerical results

Universidade do Minho

	Gradient version					Approximate descent direction version			
	$F.O.$	N_{afe}	N_{age}	f_a^*	f_{best}	$F.O.$	N_{afe}	f_a^*	f_{best}
17	0	10000000	177	4,652E+03	2,393E+03	40	10005589	2,203E-01	1,327E-01
18	100	10000000	1850	-9,160E+00	-1,026E+01	100	10004066	-1,052E+01	-1,052E+01
19	100	10000000	2126	-7,801E+00	-8,760E+00	100	10003906	-1,012E+01	-1,014E+01
20	100	10000000	1909	-9,401E+00	-9,997E+00	100	10004069	-1,037E+01	-1,039E+01
21	0	3600000	335	-1,024E+02	-1,648E+02	60	3600999	-1,867E+02	-1,867E+02
22	100	1366222	973	-4,075E-01	-4,075E-01	100	3600804	-4,075E-01	-4,075E-01
23	100	3600000	570	-1,806E+01	-1,806E+01	100	3600902	-1,806E+01	-1,806E+01
24	100	3600000	194	-2,278E+02	-2,278E+02	100	3601003	-2,278E+02	-2,278E+02
25	100	3600000	167	-2,429E+03	-2,429E+03	100	3601160	-2,429E+03	-2,429E+03
26	90	3600000	81	-2,477E+04	-2,478E+04	100	3601278	-2,478E+04	-2,478E+04
27	10	3600000	58	1,607E+05	-2,436E+05	100	3601418	-2,493E+05	-2,493E+05
28	0	10000000	141	4,470E+02	3,102E+01	60	10009759	3,977E-02	2,506E-02
29	0	10000000	135	1,289E+05	7,935E+02	0	10016905	3,633E-01	2,404E-01
30	100	1433664	16314	8,325E-112	0,000E+00	100	3601264	4,987E-07	4,464E-08
31	100	10000000	313	1,997E-13	2,780E-21	100	10005221	2,231E-04	6,612E-05
32	40	10000000	160	8,338E+00	3,031E-04	100	10006065	2,005E-03	1,186E-03

Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems
- Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima
- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.
- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.
- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.
- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.

Universidade do Minho

Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems
- Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima
- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.
- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.
- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.
- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.

Universidade do Minho

Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems
- Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima
- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.
- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.
- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.
- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.

Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems
- Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima
- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.
- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.
- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.
- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.

Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems
- Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima
- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.
- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.
- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.
- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.

Universidade do Minho

Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems
- Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima
- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.
- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.
- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.
- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.

Universidade do Minho

The end

email: aivaz@dps.uminho.pt

emgpf@dps.uminho.pt

Web http://www.norg.uminho.pt/