Matlab Realization of Parallel Random Rank Roam Method

Jaromír Kukal, Ivan Křivý, Dana Majerová*

ICT Prague, Department of Computing and Control Engineering

*University of Ostrava, Department of Informatics

Abstract. *The paper is devoted to sub–optimization technique in integer spaces. The Random Rank Roam method is introduced and its parallel form was realized as a set of functions. The Parallel 3R is a tool for sub–optimum solution finding.*

Keywords: *global optimization, sub–optimum, stochastic process, parallelization, integer space*

1 Introduction

The Random Rank Roam method (3R) [6, 7] arises from a criticism of simulated annealing (see Dekkers, Aarts and Fox [1, 3]) in a discrete space in the presence of a priori knowledge of an acceptable sub–optimum value. In this case the capture in the sub–optimum state is possible without decreasing temperature. Using the rank selection as in Winston [8], the new method is inspirable in genetic optimization (see Goldberg [5]), when the use of the rank value instead of the fitness function value improves the properties of the optimization process. The original method [6, 7] was generalized to have the better performance in the sub–optimum searching.

2 Constrained integer system

Let $n \in \mathbf{N}$ be a dimension of the system. Let $\overrightarrow{l}, \overrightarrow{u} \in \mathbf{Z}^n$ be lower and upper bound vectors satisfying conditions $l_k < u_k$ for $1 \leq k \leq n$. Then the **constrained integer system** is

$$\mathcal{S} = \{ \overrightarrow{x} \in \mathbf{Z}^n \mid l_k \leq x_k \leq u_k, 1 \leq k \leq n \}.$$

Any vector $\overrightarrow{x}_i \in \mathcal{S}$ is said to be a **state** E_i of the constrained integer system $\mathcal{S}$.

3 Sub–optimization process

Let $f : \mathbf{Z}^n \to \mathbf{R}$ be a real objective function of integer variables defined for all states $\overrightarrow{x}_i \in \mathcal{S}$. The **state value** is defined simply as

$$f_i = f(\overrightarrow{x}_i) = f(E_i).$$

Let f^* be a given real value satisfying condition

$$f^* \geq \min_{k=1,\ldots,s} f_k.$$

Then the **sub–optimum state** (minimum) is defined as a state E_i satisfying $f_i \leq f^*$. The value f^* decomposes the constrained integer system $\mathcal{S}$ into two disjoint subsystems: $\mathcal{S}_A$ containing $s_A > 0$ states and $\mathcal{S}_T = \mathcal{S} \setminus \mathcal{S}_A$, which contains $s_T = s - s_A \geq 0$ states. If f^* is equal to the minimum value of objective function f on $\mathcal{S}$, all the states of $\mathcal{S}_A$ are optimal. If $\mathcal{S}_A$ is the sub–optimum subsystem of $\mathcal{S}$, then the **sub–optimization task** (SOT) is defined as the triplet $< \mathcal{S}, f, f^* >$ or rather as a problem of finding any $E_i \in \mathcal{S}_A$. Let $< \mathcal{S}, f, f^* >$ be the sub–optimization task with $\mathcal{S}_A$ and $\mathcal{S}_T$. Let $0 \le p_{ij} \le 1$ be a probability of the transfer from $E_i \in \mathcal{S}$ to $E_j \in \mathcal{S}$ in one step. Let

$$\sum_{j=1}^{s} p_{ij} = 1, \ 1 \le i \le s.$$

The **sub–optimization process** (SOP) is defined by the rules:

(i) $E_i \in \mathcal{S}_A \Rightarrow p_{ii} = 1$,

(ii) $E_i \in \mathcal{S}_T \Rightarrow p_{ij} > 0$.

It can be represented as the quadruplet $< \mathcal{S}, f, f^*, P >$ where $P = \{p_{ij}\}_{1 \le i,j \le s}$. It is evident that the sub–optimization process is defined as a kind of the Markov chain (see Feller [2]).

Theorem 1. The sub–optimization process is the Markov chain with stochastic matrix $P = \{p_{ij}\}_{1 \le i,j \le s}$. $\mathcal{S}_A$ represents a set of its absorption states and $\mathcal{S}_T$ a set of its transient ones.

Theorem 2. The sub–optimization process converges in probability to the sub–optimum state set $\mathcal{S}_A$.

4 Mutation operator

Let $n \in \mathbf{N}$ be the system dimension and $\overrightarrow{l}, \overrightarrow{u} \in \mathbf{Z}^n$ the lower and upper bound vectors defined above. Let $\overrightarrow{x} \in \mathbf{Z}^n$ be a state of the constrained integer system. Let $\overrightarrow{g} \in (\mathbf{R}^+)^n$ be a vector of positive gains. Let $\overrightarrow{rnd} \in (0,1)^n$ be a continuous random vector variable with uniform distribution. Let $\phi : \mathbf{R} \to \mathbf{Z}$, $\mu : (0,1) \to \mathbf{R}$, and $\lambda_k : \mathbf{Z} \to \{l_k, l_k+1, \ldots, u_k\}$ for $1 \le k \le n$ be functions satisfying

$$\forall \xi \in \mathbf{R} : \lfloor \xi \rfloor \le \phi(\xi) \le \lceil \xi \rceil,$$

$$\mu(\xi) \text{ is continuous increasing function on } (0,1),$$

$$\forall \xi \in (0,1) : \mu(\xi) = -\mu(1-\xi),$$

$$\lim_{\xi \to 1-} \mu(\xi) = +\infty,$$

$$\forall k, q \in \mathbf{Z}, 1 \le k \le n, l_k \le q \le u_k : \lambda_k(q) = q.$$

Then the **mutation** is defined as an operator producing a new state $\overrightarrow{x}^* \in \mathbf{Z}^n$ by the rule

$$x_k^* = \lambda_k(\phi(x_k + g_k \mu(rnd_k))) \tag{1}$$

where $1 \le k \le n$. The mutation operator is used to generate a certain number of new states close to the starting state $\overrightarrow{x} \in \mathbf{Z}^n$.

5 Rank selection operator

Let $< \mathcal{S}, f, f^*, P >$ be a sub–optimization process. Let $\mathcal{S}$ be the constrained integer system of dimension $n > 1$. The **population** $\mathcal{P}$ is defined as any random subset of $\mathcal{S}$ consisting of $r > 1$ unique states. Let $\mathcal{P}$ be any population of size r. The **ordered population** $\mathcal{O}$ is defined as the r-tuple $(E_{k_1}, E_{k_2}, \ldots, E_{k_r})$ formed from the population $\mathcal{P}$, when $f_{k_j} \le f_{k_{j+1}}$ for all $1 \le j < r$. For the sake of clarity the ordered population can be rewritten as $(E_{(1)}, E_{(2)}, \ldots, E_{(r)})$. Let

$T > 0$ be a real parameter called the **rank temperature**. Let $rnd \in (0,1)$ be a continuous uniformly distributed random variable. Let $\nu : [0,1) \to \mathbf{R}_0^+$ be a continuous increasing function satisfying

$$\nu(0) = 0, \lim_{\xi \to 1-} \nu(\xi) = +\infty.$$

Let $\mathcal{O}$ be any ordered population of size r. Let $c \in \mathbf{N}, 1 \leq c \leq r$, be an integer-valued random variable generated by repeating the rule

$$c = \lceil T\nu(rnd) \rceil \text{ until } c \leq r.$$

Let E_j be a selected state. Then the **rank selection operator** is defined by the following rules:

$$\text{(i)} \quad f_{k_1} \leq f^* \Rightarrow E_j = E_{(1)}, \tag{2}$$

$$\text{(ii)} \quad f_{k_1} > f^* \Rightarrow E_j = E_{(c)}. \tag{3}$$

It is evident that the operator enables to select the state E_j from an ordered population $\mathcal{O}$.

6 Generalized 3R method

Let $< \mathcal{S}, f, f^*, P >$ be a sub–optimization process. Let $\mathcal{S}$ be the constrained integer system with dimension $n > 1$. Let $E_i \in \mathcal{S}_T$ be a transient state. Let $\mathcal{O}_i$ be any ordered population of size r produced by the repeated mutation of the state E_i according to (1). Then the **Generalized 3R method** is defined as the rank selection of a state E_j from the ordered population $\mathcal{O}_i$ using the rules (2) and (3). The generalized 3R method is designed to be sub–optimization process strategy. Applying generalized 3R method to any sub–optimization task, the sub–optimization process is obtained. Then the convergence to any sub–optimum state is guaranteed.

Theorem 3. When applied to any sub–optimization task the Generalized 3R method generates the sub–optimization process.

One step of Generalized 3R method can be represented in Pseudo–Pascal as

```
function G3RStep(E:state;r:integer;T:real):state;
    var
        $E_{new}$:state;
        $f_{new}$:real;
        $\mathcal{O}$ :ordered population;
        k:integer;
    begin
        MakeEmpty($\mathcal{O}$);
        for k:=1 to r do begin
            repeat
                $E_{new}$:=Mutation(E)
            until $E_{new} \notin \mathcal{O}$;
            $f_{new} = f(E_{new})$;
            Insert($\mathcal{O}$,$E_{new}$,$f_{new}$);
        end;
        G3RStep:=RankSelection($\mathcal{O}$,T);
    end;
```

The Generalized 3R method is then

```
function G3R(r:integer;T:real):state;
    var
        E:state;
    begin
        E:=RandomState;
        while $E \in \mathcal{S}_T$ do
            E:=G3RStep(E,r,T);
        G3R:=E;
    end;
```

7 Recommended design of Generalized 3R

The mutation and rank selection operators can be realized using functions

$$\phi(\xi) = \lfloor \xi + 1/2 \rfloor,$$

$$\nu(\xi) = \tan \pi\xi/2,$$

$$\mu(\xi) = \tan \pi(\xi - 1/2),$$

$$\lambda_k(q) = q + Q(u_k - l_k + 1)$$

where

$$Q = \lfloor \frac{u_k - q}{u_k - l_k + 1} \rfloor.$$

It is easy to verify the functions have the necessary properties. There is another possibility how to improve the Generalized 3R method using the multiprocessor computing system with $r > 1$ processors. Then any processor evaluates the objective function in unique state within one step of Generalized 3R method. It results the r times increasing of computation speed. The optimum size of ordered population is equal to the number of parallel processors in this case.

8 SOP parallelization

It is possible the Generalized 3R method doesn't reach any sub–optimum state when the number of objective function evaluations is constrained. It is typical when the NP–complete problem (Garey, Johnson) [4] is solved for large dimension n, the objective function is too complex, the computing system is slow or the computing time is restricted. This effect can be generalized to any sub–optimization process. The parallelization of the sub–optimization process can improve the reliability of sub–optimum finding in these cases.

Let $< \mathcal{S}, f, f^*, P >$ be given sub–optimization process. Let $r \in \mathbf{N}$ be a number of evaluation of $f(\overrightarrow{x})$ in every step of the SOP. Let $H \in \mathbf{N}$ be given number of independent parallel realizations of the SOP. Then the vector $\overrightarrow{E} = (E_{i_1}, \ldots, E_{i_H}) \in \mathcal{S}^H$ is called **multi–state**. **Initial multi–state**is taken at random and denoted as $\overrightarrow{E}_0$. **Terminal multi–state** $\overrightarrow{E}_m$ is defined as a multi–state reached in m–th step and satisfying the condition

$$\exists k \in \{1, \ldots, H\} : E_{i_k} \in \mathcal{S}_A.$$

Then **parallel sub–optimization process** (PSOP) $< \mathcal{S}, f, f^*, P, H >$ is defined as a Markov chain on $\mathcal{S}^H$ generating $\overrightarrow{E}_{new}$ from $\overrightarrow{E}$ by independent SOP's using the rule

$$E_{i_k}^{new} = SOP(E_{i_k})$$

for $1 \leq k \leq H$. Then the PSOP generates a multi–state sequence $\overrightarrow{E}_0, \ldots, \overrightarrow{E}_m$. It is easy to recognize the PSOP with $H = 1$ is just the original SOP. Every step of PSOP includes Hr evaluations of the objective function and then reaching the terminal multi–state includes Hrm function evaluations. The question about ideal value of H minimizing the number of evaluations within PSOP is also under the scope of this paper.

Theorem 4. Let $< \mathcal{S}, f, f^*, P >$ be given SOP. Let $F(k)$ be distribution function of its step number. Let $< \mathcal{S}, f, f^*, P, H >$ be adjoined PSOP. Let $F^*(k)$ be distribution function of its step number. Then $F^*(k) = 1 - (1 - F(k))^H$.

Theorem 5. Let $< \mathcal{S}, f, f^*, P, H >$ be given PSOP. Let $F(k)$ be distribution function of its SOP step number. Let $F^{(-1)} : [0; 1) \to \mathbf{N}_0$ be pseudo–inversion of F satisfying $F^{(-1)}(p) = \min$ with respect to $F(F^{(-1)}(p)) \geq p$ for all $p \in [0; 1)$. Let m^* be given constrain of the PSOP step number. Let $0 < \alpha < 1$ be given upper bound of the PSOP failure probability after m^* parallel steps. Then $m^* \geq F^{(-1)}(1 - \alpha^{1/H})$ and the minimum number of objective function evaluations is

$$N_e = HrF^{(-1)}(1 - \alpha^{1/H}).$$

Let $< \mathcal{S}, f, f^*, P >$ be given SOP. Let $F(k)$ be distribution function of its step number. Let $< \mathcal{S}, f, f^*, P, H >$ be adjoined PSOP. Let $F^*(k)$ be distribution function of its step number. Let $0 < \alpha < 1$ be given upper bound of the PSOP failure probability. Finding

$$H^* = arg \min_{H \in \mathbf{N}} HF^{(-1)}(1 - \alpha^{1/H})$$

is called **optimum parallelization** of SOP. Let $\widehat{F}(k)$ be an approximation of $F(k)$. Finding

$$\widehat{H} = arg \min_{H \in \mathbf{N}} H\widehat{F}^{(-1)}(1 - \alpha^{1/H})$$

is called **approximated parallelization** of SOP.

Theorem 6. Let $0 < \alpha, p_{fail} < 1$. Let

$$\lim_{k \to \infty} F(k) \leq 1 - p_{fail}.$$

Then

$$H^* \geq \frac{\ln \alpha}{\ln p_{fail}}.$$

Let $0 < p_{fail} < 1$ be failure probability of given SOP. Let $0 < \alpha < 1$ be given upper bound of PSOP failure probability. Then **heuristic parallelization** is defined as using

$$H^+ = \lceil \frac{\ln \alpha}{\ln p_{fail}} \rceil$$

within the PSOP. Then the poor knowledge of statistical properties of the SOP enables to make first optimistic experiments with parallelization. There is necessary to use parametric statistical models of $F(k)$ for future detail analysis.

9 Parallel 3R method

The general principle of SOP parallelization can be applied to any sub–optimization process. When applied to sub–optimization task solving the Generalized 3R method can be paralleled, too. The result is the Parallel 3R method. Its realization on a single processor can be writen as

```
function P3R(H,r:integer;T:real):state;
    var
        E:array [1..H] of state;
        i:integer;
    begin
        for i:=1 to H do
            E:=RandomState;
        while AllTransient(E) do
            for i:=1 to H do
                E[i]:=G3RStep(E[i],r,T);
        for i:=1 to H do
            if E[i] $\in \mathcal{S}_A$ then
                P3R:=E[i];
    end;
```

Parallel 3R method comes to Generalized 3R method for $H = 1$. Parallel 3R method can be also realized on any multiprocessor system in pure parallel style using Hr processors for the objective function evaluations. It results the Hr times increasing of the computation speed.

10 Matlab realization of Parallel 3R method

The Parallel 3R method can be realized on single processor in Matlab as a general tool for sub–optimization in constrained integer spaces. The Parallel 3R library consists of four general functions

```
xsub,fsub=Parallel3R(l,u,fobj,fopt,H,r,T,g);
xnew=Step3R(x,l,u,fobj,fopt,r,T,g);
xnew=RankSelect3R(X,f,T);
xnew=Mutation3R(x,l,u,g);
```

The objective function can have any name and is called as

```
F=ObjFunction(x);
```

where the input and output parameters are described as

```
x ...... integer state vector (1,n)
xnew ... new integer state vector (1,n)
l ...... lower bound vector (1,n)
u ...... upper bound vector (1,n)
fobj ... name of the objective function as string
fopt ... sub-optimization threshold value f* as scalar
xsub ... sub-optimum solution as vector (1,n)
fsub ... sub-optimum value as scalar
H ...... number of parallel realizations as positive integer
r ...... ordered population size as positive integer
```

```
T ...... rank temperature as positive number
g ...... gain vector of positive numbers (1,n)
X ...... matrix of population (r,n)
f ...... vector of population values (r,1)
F ...... objective function value as scalar
```

The Parallel3R function calls many times Step3R one while Step3R function calls r-times the objective function, many times the Mutation3R function and onetime the RankSelect3R one.

11 Conclusions

The Parallel 3R method is an effective tool for sub–optimum state finding. The recommended values of parameters are $1/8 \leq T \leq 4,\ 3 \leq r \leq 10,\ 1/100 \leq g_k \leq 2$ while the parameter H depends on the complexity of solved sub–optimization task.

References

[1] Dekkers A., Aarts E.*Global Optimization and Simulated Annealing.* Mathematical Programming, p. 50–367,1991.

[2] Feller W. *An Introduction to Probability Theory and Its Applications.* John Wiley & Sons, New York, 1957.

[3] Fox B.L. *Simulated annealing: Folklore, Facts and Directions.* October, 1994.

[4] Garey M.R., Johnson D.S. *Computers and Intractability: a Guide to the Theory of NP–Completeness.* Freeman, San Francisco, 1979.

[5] Goldberg D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley, Reading, 1989.

[6] Kukal J. *Stochastic Global Optimization Techniques in Artificial Intelligence.* Ph.D. Thesis. Ostrava: University of Ostrava, 2000.

[7] Kukal J., Křivý I. *Random Rank Roam Method.* In it Mendel 2000. 6th International Conference on Soft Computing. pp. 200–205, Brno, 2000.

[8] Winston P.H. *Artificial Intelligence.* Addison Wesley, Reading, 1992.

Contact

Jaromír Kukal, Dana Majerová

INSTITUTE OF CHEMICAL TECHNOLOGY, Prague
Department of Computing and Control Engineering
Technická 5, 166 28 Prague 6 Dejvice
Phone: 420-2-2435 4212, fax: 420-2-2435 5053
E-mails: {Jaromir.Kukal, Dana.Majerova}@vscht.cz
WWW addresses: http://www.vscht.cz/ (university),
http://uprt.vscht.cz/ (department),
http://phobos.vscht.cz/ (research group)