MULTI-RESOLUTION OPTIMISATION: APPLICATION OF META-HEURISTIC IN FUNCTION REMODELLING

Cheng Wai Kheng, Siang Yew Chong, Andrzej Bargiela
School of Computer Science,
The University of Nottingham, Malaysia Campus,
Jalan Broga, Semenyih, 43500, Selangor, Malaysia
Email: {cheng-wai.kheng, siang-yew.chong, andrzej.bargiela}@nottingham.edu.my

KEYWORDS

ABSTRACT
Multi-resolution approach to optimisation involving remodelling of objective function has great received interests lately. There are two different goals of remodelling for the objective function. First, the reduction of computational complexity of the objective functions in order to accelerate the searching process. Second, the removal of local optima in the objective functions to avoid premature convergence. Most of the approach shares the problem: the setting of the smoothness pressure value over the objective function. We proposed a meta-heuristic approach to determine the suitable smoothness pressure value throughout the searching process. The results show that meta-heuristic has significant improvement compared to state-of-the-art Classic Evolutionary Programming (CEP) and Fast Evolutionary Programming (FEP).

INTRODUCTION
Continuous Optimisation is a non-linear programming problem that involves finding the optimal solution in a given solution space. Often, the form of optimisation is defined as an objective function, which determines the fitness of given solution (Goldberg 1989; Lange 2004). A solution is defined as a vector or a set of real values. If the problem is bounded, the values are restricted in the lower bound and upper bound of the problem. Such as follow:

\[ \min f(x) \text{ such that } \forall x', x, f(x') > f(x) \]

where \( x, x' \in X \) and \( X \subset \mathbb{R}^n \)

The application of Evolutionary Algorithm (EA) in optimisation received attention due to their stochastic characteristic and capability of escaping local optima in the solution space (Goldberg 1989). Two issues that always surrounded by EA are the convergence speed and the quality of converged solution. Several algorithms that are based on EA have been proposed. These includes Memetic Algorithm (Krasnogor 2002), Cooperative Co-Evolutionary Algorithm (Popovici and Jong 2006). Other researches include improving the three major components of EA, which are the problem representation, selection and variation methods. Recently, variation method with higher probabilities of escaping local optima have been studied, such as Cauchy Mutation (Xin et al. 1999), Levy Mutation (Chang-Yong and Xin 2004), both producing promising results over benchmark testing functions.

On the other hand, improvement on problem representation is the remodelling of objective function. There are two different types of remodelling the objective function. First, the reduction of computational complexity of the objective functions in order to accelerate the searching process. Several regression techniques have been proposed, such as polynomial regression (Branke and Schmidt 2005), Gaussian/Kriging regression (Ratle 1998, 1999; Buche et al. 2005; Nakayama et al. 2001), these techniques work under the condition that the remodelled function provides correct smoothness and certain degree of similarity compared to the objective function. As concluded by Liang et al, the Quadratic Approximation model performs worst than without remodelling if the problem does not carry quadratic characteristic (Liang et al. 2000). Gaussian Smoothing Kernel shares the same problem which the kernel size affects the smoothness of the remodelled function. Dekun Yang et al proposed a coarse-to-fine mechanism, where the kernel size reduces according to the searching progress. Bernardetta Addis et al discussed the same problem in (Addis et al. 2005) where Gaussian Smoothing Kernel is used together with numerical optimisation. In this paper, we further the investigation of the effect of Gaussian Smoothing
Kernel in optimisation, and proposed a metaheuristic approach in choosing the kernel size which reacts according to the current population.

BACKGROUND STUDY

Noise reduction is always an interesting issue in signal/image processing. The main purpose of noise reduction is to filter the unwanted signals that are caused by external factors. Local optima in optimisation share this characteristic, where the solution’s fitness does not reflect the actual distance from optimal solution. To smooth continuous signals, a convolution transformation has to be done. As depicted in (Yang and Flockton 1995), the convolution transformation, or the approximation of function $f$ is the integration of multiplication of smoothing kernel and the function. Let $F_\beta$ be the approximated function with $\beta$ as the smoothness pressure.

$$F_\beta(x) = \int_{y \in \mathbb{R}} h_\beta(y) f(x-y) dy$$

(2)

where $h_\beta(y)$ satisfied $\lim_{\beta \to 0} h_\beta(y) = \delta(y)$

Gaussian function satisfied condition in (2) and is given as below:

$$h_\beta(x) = \frac{1}{(2\pi)^{N/2} \beta^N} \exp(-x^T x / 2\beta^2)$$

(3)

The $\beta$ and $N$ are the kernel size and number of dimension respectively. That is, the higher the $\beta$, the smoother the approximated function. As Evolutionary Algorithm treats the optimisation as a black box searching process, the convolution of the function itself is prohibited. However, statistical learning suggested that if the sample size is large enough, then the convolution yields the similar result as the expected value (mean) of the fitness value of neighbour solutions. These solutions are sampled randomly according to the probability of Gaussian distribution, as depicted as $y$ in (4).

$$F_\beta(x) = E_{\beta} \{ f(x-y) \}$$

(4)

And the expected value of these neighbour solutions is its sample mean:

$$F_\beta(x) = \frac{1}{M} \sum_{i=1}^{M} f(x-y_i)$$

(5)

with $y_i$ is randomly selected according to Gaussian distribution with $\beta$ as the standard deviation. (2) to (5) are summarized from Dekun Yang et al (Yang and Flockton 1995). As discussed above, the $\beta$ is the smoothness pressure of the approximated function. The $\beta$ has to be carefully selected because if it is too small, the approximated function does not remove the local optima, whereas if it is too high, it removes the global optimum as well (Yang and Flockton 1995). In (Yang and Flockton 1995), a coarse-to-fine mechanism is used, where $\beta$ is arbitrary set to 3.0 initially and reduced 30% every eighty generation. First, the coarse-to-fine mechanism does not reflect the underlying optimisation problem. Second, the approximated function $F_\beta$ does not guarantee to have the same global optimum as original fitness function $F$, this is shown in figure 1, where $\beta=3.0$ and 2.0 shifted the global optimum of the function. This is referred as “curse of uncertainty”, a term was coined in (Ong et al. 2006). It indicates that the improper approximation causes the local search returns non-local-optimum solution in $F$, but the solution appears to be local optimum in $F_\beta$. In fact, the smoothed function only provides the “area of interest” where there no local optima. Consider figure 2 and 3. Figure 2 suggested that the smooth function removed local optima while the global optimum remains. However, the finer interval in figure 3 shows that the global optimum in objective function remains in single point while the approximated function does not carry any optimum. Thus, the approximated function should only provides the “area of interest” while the actual searching process continues with solutions found in the approximated function.
METHODOLOGY

From the hyperheuristic/metaheuristic point of view, the approximated functions with different smoothness are different heuristics for solving the problem. Choosing the good approximation may result in significant improvement on the algorithm, which includes faster convergence speed and better converged solution. A heuristic is informal action used in solving problem, often is cheap in computation, is easy to implement and yet a natural way of solving the problem(Cowling et al. 2002). A hyperheuristic is a heuristic that chooses low level heuristic independently from problem, such as CPU execute cycles, improvement of a consecutive heuristic, etc(Kendall and Soubeiga 2002; Cowling et al. 2002). Metaheuristic is a hyperheuristic but the choosing function is often problem dependent (Cowling et al. 2002), such as pre-clustering of class of problem and improving a consecutive heuristic, etc(Ong and Keane 2004).

Approximated function $F_{\beta}$ should be balanced between the smoothness and the similarity. Smoothness is referred to the number of local optima in the searching space. It can be represented by the inverse of roughness. However, there is no other way to quantify local optima in the function except by finding all of them. Fortunately, the fluctuation of the function also reflects the number of local optima. The fluctuation of the function is associated with the standard deviation of itself, $\sigma(F)$.

\[
\text{roughness of } F \propto \text{fluctuation of } F \propto \sigma(F)
\]

\[
\mu = \frac{1}{N} \sum_{i=1}^{N} F_i
\]

\[
\sigma(F) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (F_i - \mu)^2}
\]

Similar to (2), the integration is prohibited, and can be replaced with unbiased estimator with uniform sampling:

\[
\tilde{\mu} = \frac{1}{N} \sum_{i=1}^{N} s_i
\]

\[
\tilde{\sigma}(F) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (s_i - \tilde{\mu})^2}
\]

Similarity is referred here to the difference between the global optimum in $F_{\beta}$ and $F$. Also, this cannot be assessed as the searching process does not know the actual global optimum in the searching space. However, the area difference between $F_{\beta}$ and $F$ may reflect the similarity as well. Thus, the comparison of the area of the underlying function is defined as $\lambda(F)$, which is normalised from root mean square error "rmse" from (Ong et al. 2006):

\[
\text{area difference } \times \lambda(F_p) = \frac{\| F_p(x) - F(x) \|}{\| F(x) \|}
\]

For $\beta \in \mathbb{R}^+$

Followed by an unbiased estimator with uniform sampling:

\[
\lambda(F_p) = \frac{\sum_{i=1}^{N} F_p(x_i) - F(x_i)}{\sum_{i=1}^{N} F(x_i)}
\]

In order for (9-10), (12) to estimate the roughness and area difference between the approximated and objective function accurately, sample size $N$ must be very large. Fortunately, the sample size can be reduced if the quantifying process only takes place for the current searching area, which are the solutions in the current population. Let $p$ be the current population, and $I_{\text{best}}$ and $I_{\text{worst}}$ be the best and worst solution in population $p$. The current searching space is defined as $\gamma$, with abbreviation $\gamma(i)$ indicate value of $i^{th}$ element:

\[
\gamma(i) = \{ x \in \mathbb{R} | I_{\text{best}}(i) < x < I_{\text{worst}}(i) \}
\]

The uniform sampling processes for (8) and (12) are performed within $\gamma$. The empirical model for the metaheuristic in choosing $F_{\beta}$ out of a predefined set $\beta=\{\beta_1, \beta_2, \ldots, \beta_n\}$ is the $\beta$ value that produce the minimum output of $M(\beta)$:

\[
M(\beta) = a \sigma(F_p) + (1 - a) \lambda(F_p)
\]

Where $a$ is the weight associate with the roughness and area difference for each approximated function $F_{\beta}$. In this paper we select $a$ value as 0.3. The authors are aware that $M(\beta)$ is a multi-objective optimisation problem. We avoid using multi-objective optimisation in this event due to the demonstration here is not about to choose the exact $\beta$ but to estimate the good balance between smoothness and similarity.

The proposed algorithm is outlined in pseudo code in Fig 4. The procedure starts by randomly generates solutions into the population. To determine which approximation function $F_{\beta}$ to use, the algorithm has to choose the suitable $\beta$, which $M(\beta)$ evaluates all beta values and returns the suitable $\beta$. There are two phase in the searching process. The first phase search in $F_{\beta}$ and when it convergence condition is met, the process continues on the second phase, which searches in $F$. The convergence condition we experimented is the number of generation without improvement, and it is set to two hundred generations.
Define Population as pop  
Populate pop  
Initialize Beta = \{3.0, 2.0, 1.0, 0.5, 0.3, 0.1\}  
Initialize generation=1  
Choose B from Beta based on M(\(\bar{\eta}\))  
Evaluate pop based on F  
While(termination condition)  
DO  
set improvement=0  
While(i<population size)  
DO  
Select individual i from pop  
offspring = GaussianMutation(i, mutation rate)  
IF(offspring fitness better than i)  
improvement+1  
ADD offspring into pop  
ENDIF  
ENDIF  
IF converged on F  
ELSE  
evaluate pop based on F  
ENDIF  
IF (generation%20 =0)  
DO  
IF not converged on F  
Set B based on M(\(\bar{\eta}\))  
evaluate pop based on F  
ELSE  
Set mutation rate = mutation rate*1.2  
ELSE  
Set mutation rate = mutation rate*0.8  
ENDIF  
ENDIF  
Truncate pop  
ENDDO  
Return best solution of pop  

EXPERIMENTAL SETTING

For the purpose of comparison, we implemented the metaheuristic and smoothing in Evolutionary Algorithm with 20% adaptive mutation rule, called EA+S. this algorithm is compared with Classic Evolutionary Programming (CEP), Fast Evolutionary Programming (FEP), and Evolutionary Algorithm with 20% adaptive mutation rule (EA). For simplicity, selection method implemented in these algorithms is truncation. EA+S revise the \(\beta\) value every twenty generations. For EA and EA+S, the adaptive mutation rate is also revised every twenty generations. There are totally ten benchmark functions tested with these experiments settings. Table 1 outlines all benchmark functions. F\(_i\) to \(F_{10}\) are taken from (Xin et al. 1999) while \(F_{10}\) is originally from (Yang and Flockton 1995).

To minimize the bias toward different type of problems between EA and EP, instead of using uniform selection for parents, each parent in EA produces exactly one offspring, as suggested in CEP and FEP (Xin et al. 1999). The only difference between EA and EP in this case is the adaptive mutation mechanism. For EA+S, the searching process is divided into two phases. In the first phase, solutions in population are evaluated with approximated function \(F_{\bar{\eta}}\). After the population has converged on \(F_{\bar{\eta}}\), the algorithm then continues the searching with the original function F.

Table 1: The Following Ten Benchmark Functions Have Been Used in Evaluating Effect of Smoothing. N Indicate The Number Of Dimension Of The Problem, S and \(f_{\text{min}}\) Indicates Function Range and Function Minimum respectively.

<table>
<thead>
<tr>
<th>Test Function</th>
<th>n</th>
<th>S</th>
<th>(f_{\text{min}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_1(x) = \sum_{i=1}^{n} x_i^2)</td>
<td>30</td>
<td>[-100,100](^n)</td>
<td>0</td>
</tr>
<tr>
<td>(f_2(x) = \max_i</td>
<td>x_i</td>
<td>, i \leq n)</td>
<td>30</td>
</tr>
<tr>
<td>(f_3(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2])</td>
<td>30</td>
<td>[-30,30](^n)</td>
<td>0</td>
</tr>
<tr>
<td>(f_4(x) = \sum_{i=1}^{n} [1 - \cos(\pi x_i)])</td>
<td>30</td>
<td>[-500,500](^n)</td>
<td>-12569.5</td>
</tr>
<tr>
<td>(f_5(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_1^2 + 4x_2^4)</td>
<td>2</td>
<td>[-5,5](^n)</td>
<td>-1.0312685</td>
</tr>
<tr>
<td>(f_6(x) = -20 \exp(-0.2 \sum_{i=1}^{n} x_i^2) \sum_{i=1}^{n} \cos(\sum_{j=1}^{i-1} x_j) + 1)</td>
<td>30</td>
<td>[-32,32](^n)</td>
<td>0</td>
</tr>
<tr>
<td>(f_7(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^4 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1)</td>
<td>30</td>
<td>[-600,600](^n)</td>
<td>0</td>
</tr>
<tr>
<td>(f_8(x) = \sum_{i=1}^{n} (x_i + 0.5)^2)</td>
<td>30</td>
<td>[-100,100](^n)</td>
<td>0</td>
</tr>
<tr>
<td>(f_9(x) = a(x_i - bx_i^2 + cx_i - d)^2 + e(1-f)\cos(\eta x_i) + c)</td>
<td>2</td>
<td>[0.15](^n)</td>
<td>0.397887</td>
</tr>
<tr>
<td>(a = 1, b = 5)</td>
<td>5</td>
<td>(c = \frac{5}{6})</td>
<td>0</td>
</tr>
<tr>
<td>(f_{10}(x) = 0 + \sum_{i=1}^{n} x_i^2 - \cos(2\pi x_i))</td>
<td>30</td>
<td>[-5.12,5.12](^n)</td>
<td>0</td>
</tr>
</tbody>
</table>

This is to avoid the situation where the approximated function shifted the global optimum as shown in Fig 1 and 3. For (5), the number of samples used here is the same as suggested in (Yang and Flockton 1995), which is 10. The sample size for (10) and (12) are 500 for both. As the sample size increases, (5), (10) and (12) becomes more accurate. Further investigation of the minimum sample size is required in the future.
The population size for all algorithms is set to 100, while the initial mutation rate is set to 3.0. The Gaussian random is time seeded. To avoid having different random sequence between algorithms, random sequence for (3), (10), and (12) are separated from mutation. Twenty individual runs for each algorithm is tested with all benchmark problem in table 1.

RESULT AND DISCUSSION

Recall that the purpose of smoothing in optimisation is to “pull” the population into the “area of interest” where the local optima do not exist. The ideal population that converged on approximated function should always within the “area of interest”. However, the results suggested that it may not be this case. For EA+S, chances of “pulling” into the area are relatively high for F1, F3, F5, F6, F8 and F10. Consider figure 5, 6, and 7, majority of the individual runs successfully reach the area of interest, and converged on the global optimum in the second phase. EA+S suggested a huge improvement over these functions compared to EA. The reason we choose to compare our algorithm with CEP and FEP because both of these algorithms are the state-of-the-art in Evolutionary Computation. As we compare the result in table 2, EA+S always have the better quality of solution than EA, as well as FEP. The only exceptional cases are F2 and F6. Function F2 is neither polynomial nor trigonometry, which is very hard for Gaussian to estimate the pattern of the function, whereas F6 has very sharp local optima, which requires higher \( \beta \) value. In fact, as shown in figure 8, function F6 requires at lest \( \beta=80 \) for the approximation to be smooth. Since the maximum setting for \( \beta \) is 3.0 in our experiment, we expect that EA+S perform worst compare to FEP and CEP. This raises another issue where the \( \beta \) range has to be predetermined before the searching process actually starts. Pre-processing such as performing random walk in \( \beta \) space is required.

As for the comparison of the convergence speed, figure 9 and 10 shows the significant difference between the convergence speeds for EA+S and FEP for function F6. That is, all individual runs converged to global optimum within 300 generations for EA+S whereas only one individual run converged within 700 generations for FEP. However, this may due to several factors other than choosing the correct smoothness to approximate the function. Recall that the adaptive mutation rate for EA and EP are different. EA uses single mutation rate for all dimensions of the problem, whereas in EP, each dimension of the problem has their own adaptive mutation rate.

CONCLUSION AND FUTURE WORK

Based on the empirical results of the benchmark functions in table 2 and 3, we concluded that the metaheuristic does assist in the choosing the correct smoothness, providing a compatible underlying EA. However, its compatibility with other parameters in EA is still an open question. Further studies are required in order to understand the behaviour of Gaussian approximation in EA. We will analyse different adaptive mutation mechanism to maintain the population diversity without decreasing the probability of getting into the “area of interest”. Throughout statistical learning, we may discover more meaningful information regarding the underlying problem and propose different meta-heuristic that is capable of handling various problems in the future.

<table>
<thead>
<tr>
<th>Function</th>
<th>EA</th>
<th>CEP</th>
<th>FEP</th>
<th>EA+S</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>104.05</td>
<td>14.33</td>
<td>1.57</td>
<td>0.31</td>
</tr>
<tr>
<td>F2</td>
<td>3.91</td>
<td>0.26</td>
<td>12.59</td>
<td>5.77</td>
</tr>
<tr>
<td>F3</td>
<td>18913.85</td>
<td>16422.29</td>
<td>13347.76</td>
<td>20508.87</td>
</tr>
<tr>
<td>F4</td>
<td>-7503.03</td>
<td>593.4426</td>
<td>-10587.7</td>
<td>412.3559</td>
</tr>
<tr>
<td>F5</td>
<td>-1.03147</td>
<td>0.000171</td>
<td>-1.03163</td>
<td>4.56E-16</td>
</tr>
<tr>
<td>F6</td>
<td>8.236878</td>
<td>2.199107</td>
<td>2.271381</td>
<td>1.688895</td>
</tr>
<tr>
<td>F7</td>
<td>0.327072</td>
<td>0.060595</td>
<td>0.23566</td>
<td>0.153455</td>
</tr>
<tr>
<td>F8</td>
<td>105.7</td>
<td>10.87779</td>
<td>922.3</td>
<td>917.0538</td>
</tr>
<tr>
<td>F9</td>
<td>0.397976</td>
<td>8.45E-05</td>
<td>0.397887</td>
<td>1.14E-16</td>
</tr>
<tr>
<td>F10</td>
<td>122.505</td>
<td>8.64936</td>
<td>27.78619</td>
<td>8.814827</td>
</tr>
</tbody>
</table>

*Mean Best with **BOLD** indicates that there are 95% confindence that they are statistically significant different compare to EA+S in T-Test. Mean Best with *Italic* in EA+S column indicates it is the best among others.
Fig 5. Natural Log Of Best Fitness of 20 Individual Runs of EA+S In 3500 Generations for Function F1.

Fig 6. Natural Log Of Best Fitness of 20 Individual Runs of EA+S In 3500 Generations for Function F3.

Fig 7. Natural Log of Best Fitness of 20 Individual Runs of EA+S In 3500 Generations for Function F10.

Fig 8. Single Dimension of Function F4 With y-axis as The Fitness Value Against x-axis, and Beta=50, 80.

Fig 9. Natural Log Of Best Fitness of 20 Individual Runs of FEP In 3500 Generations for Function F4.

Fig 10. Natural Log of Best Fitness of 20 Individual Runs of EA+S In 3500 Generations for Function F8.

REFERENCES


Ong, Y.-S., Z. Zhou, and D. Lim. 2006. Curse and Blessing of Uncertainty in Evolutionary Algorithm Using Approximation

In IEEE Congress on Evolutionary Computation.


AUTHORS BIOGRAPHIES

CHENG WAI KHENG is a Postgraduate Student of School of Computer Science, University of Nottingham, Malaysia Campus. His current research interests include the application of Hyper/Meta Heuristic, Evolutionary Computation and Optimisation. He received his B.Sc in Computer Science from University of Nottingham, Malaysia in 2007.

SIANG YEW CHONG is an Assistant Professor with the School of Computer Science, University of Nottingham, Malaysia Campus, a member of the Automated Scheduling, Optimization and Planning (ASAP) Research Group, University of Nottingham, UK, and an Honorary Research Fellow with the School of Computer Science, University of Birmingham, UK. He was a Research Associate with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), University of Birmingham, UK, in 2007. His major research interests include evolutionary computation, neural networks, and evolutionary game theory. Dr. Chong received the Ph.D. degree in Computer Science from the University of Birmingham, UK, in 2007, and was awarded the 2009 IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation Award for his work on theoretical frameworks for performance analysis in co-evolutionary learning.

ANDRZEJ BARGIELA is Professor and Director of Computer Science at the University of Nottingham, Malaysia Campus. He is member of the Automated Scheduling and Planning research group in the School of Computer Science at the University of Nottingham. Since 1978 he has pursued research focused on processing of uncertainty in the context of modelling and simulation of various physical and engineering systems. His current research falls under the general heading of Computational Intelligence and involve mathematical modelling, information abstraction, parallel computing, artificial intelligence, fuzzy sets and neurocomputing.