A Group-based Approach to Improve Multifactorial Evolutionary Algorithm

Jing Tang[1], **Yingke Chen**[1], **Zixuan Deng**[2], **Yanping Xiang**[2], **Colin Paul Joy**[1]
[1] Department of Computer Science & Information Systems, Teesside University, UK
[2] School of Computer Science & Engineering,
University of Electronic Science and Technology of China, China
{j.tang, y.chen, c.joy}@tees.ac.uk, deng_zxuan@foxmail.com, xiang_yanping@gmail.com,

Abstract

Multifactorial evolutionary algorithm (MFEA) exploits the parallelism of population-based evolutionary algorithm and provides an efficient way to evolve individuals for solving multiple tasks concurrently. Its efficiency is derived by implicitly transferring the genetic information among tasks. However, MFEA doesn't distinguish the information quality in the transfer compromising the algorithm performance. We propose a group-based MFEA that groups tasks of similar types and selectively transfers the genetic information only within the groups. We also develop a new selection criterion and an additional mating selection mechanism in order to strengthen the effectiveness and efficiency of the improved MFEA. We conduct the experiments in both the cross-domain and intra-domain problems.

1 Introduction

Classical evolutionary algorithms (EA) use population based search over the solution space to solve an optimization task. As a recently proposed learning paradigm, Multifactorial evolutionary algorithm (MFEA) [Gupta *et al.*, 2016] aims to solve multiple optimization tasks simultaneously in a unified search space, where an individual in the population evolves based on the factorial influences received from every task and every task could be either a single objective or multiple objective optimization problem. Considering that real world problems seldom exist in isolation, a set of parameters tuned for one task are likely to be useful for a similar optimisation. This helps circumvent challenges associated with the dimensionality curse when several tasks with multidimensional search spaces are to be solved simultaneously. MFEA leverages the genetic complementaries transferring between tasks and enhances the averaged performance of solving the tasks. Its obvious advantage in time over a single-task EA demonstrates enormous potential of responding to the rapidly expanding of cloud computing industry. Multiple optimisation tasks that are received simultaneously from multiple users need to be solved within a limited amount of time in the cloud, which stresses on the time requirement for the multitasking.

Recently, MFEA has shown great potential on solving real-world problems like complicated engineering design [Cheng *et al.*, 2017], capacitated vehicle routing Problem [Zhou *et al.*, 2016] and so on. For instance, in engineering design, a variety of possible product designs are analysed and optimised during the conceptualisation phase (before one of them is finally chosen). Speeding up the analysis/design-optimisation stage through a multitasking solver can considerably reduce the often exorbitant design time.

What MFEA differs most from the single-task EA is the existence of genetic communication between tasks. The current version of MFEA has all the tasks share the genetic information without evaluating its potential impact on solving the tasks. In this paper, we group the tasks of similar types and transfer the genetic information only within the groups. This will differentiate the genetic information transfers among tasks so that the tasks share only the genetic information with positive mutual impact. Without solving the tasks, a proper task differentiation is difficult since it is hard to know what genetic exchange will contribute to solving the task. However, individuals (in a population) that are employed to solve the tasks in MFEA will reflect the characteristics of the tasks (e.g. the landscape of objective functions of the tasks), which can be used to group the tasks. The similar tasks grant the individuals' commonality that facilitates the same task management.

Inspired by this idea, we propose group-based MFEA (GMFEA) that restricts individual behaviour within a group of similar tasks. Unlike the MFEA of giving an equal opportunity of inter-task genetic communication, GMFEA controls the genetic transfer among tasks belonging to the same group. We observe that the tasks having near global optima often choose individuals that follow similar solution paths, which is also verified in the experimental study (in Fig. 10). Subsequently multiple traits stemming from multiple tasks can exert a joint influence on the individual evolution in a positive way. Hence we group tasks from both the genotype and phenotype of individuals in a population. Genotypes consider genetic composition of individuals while phenotypes are their fitness values. The grouping could be conducted without prior knowledge about the task properties.

Introducing the grouping operator may compromise the population diversity resulting in poor algorithm performance. To mitigate this problem, we propose a new selection crite-

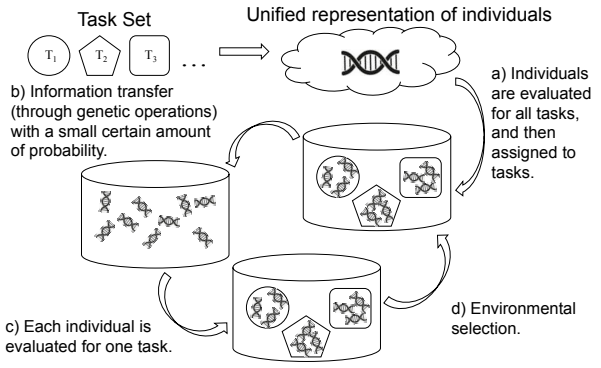


Figure 1: The MFEA main procedures.

rion that considers a tradeoff between individual fitness and population diversity. The new consideration significantly reduces the evaluation time while maintaining good performance. We summarise the main contributions below.

- We propose a group-based MFEA to control the inter-task genetic communication in a dynamic way;
- We design a new selection criterion considering both fitness and diversity, which is used in the group-based mating selection and environmental selection;
- We conduct experiments in both cross-domain and intra-domain problems to demonstrate performance of the new algorithm.

2 Background Knowledge and Related Works

Evolutionary algorithms are optimisation metaheuristic which mimics the process of natural evolution [Bäck *et al.*, 1997]. EAs have been widely used to solve corresponding optimisation problems in multiple domains, e.g, single-objective optimisation [Guo and Yang, 2015] , multi-objective optimisation [Coello, 2006; Fonseca and Fleming, 1995] and so on. Recently, inspired by the concept of 'multifactorial inheritance', Gupta *et al.* [2016] introduce multifactorial evolutionary algorithm (MFEA) that further promotes the concurrent search to optimize multiple tasks simultaneously. The MFEA splits a population into different skill groups each of which excels at a different task. The genetic material created within a particular group turns out to be useful for another task as well. Thus, in such situations, the scope for genetic transfer across tasks can potentially lead to the discovery of otherwise hard to find global optima.

Fig. 1 elaborates four main procedures of MFEA that start from a unified representation of individuals for the tasks, and then iterates individual evaluation, genetic information transfer and environmental selection. In every iteration, only promising candidates to solve one task (in the right container) survive after an environmental selection that is conducted in the bottom container. We aim to improve the basic MFEA framework in this paper.

On theoretical grounds, MFEA amplifies the power of population-based search. As genetic building blocks corresponding to different optimisation tasks are contained within a unified pool of genetic material, they get processed by the EA in parallel [Wright *et al.*, 2003]. Most importantly, this encourages the discovery and implicit transfer of useful genetic material from one task to another in an efficient manner. Moreover, as a single individual in the population may inherit genetic building blocks corresponding to multiple optimisation tasks, the analogy with multifactorial inheritance becomes more meaningful.

We shall note that MFEA deals with multiple fundamentally distinct optimisation tasks while coevolutionary algorithms [Trunfio, 2015] handle a single optimisation task. It is indeed conceivable to design a coevolutionary algorithm capable of multitasking. However, this requires several hyperparameters to be specified in the coevolution framework. Meanwhile, MFEA is more general than multi-objective optimisation (MOO). MOO deals with conflicts among competing objectives belonging to *one* given task. MFEA is to optimize each constitutive task absolutely, instead of having to establish any kind of trade-off between individual tasks.

Finally, another relevance is multitask learning that have been well studies in the machine learning field [Caruana, 1997]. The multitask learning research mainly focuses on investigation of relatedness among different tasks and knowledge transfer among the learning domains. It is not intended for solving multiple-function optimisation problems and is not rooted in the EA principle. MFEA is the state-of-art solver for solving multifactorial optimisation problems.

3 Group-based MFEA

We first present the grouping operator in the new algorithm and then develop a new selection criterion to improve the mating selection and environmental selection.

3.1 Grouping Tasks

Grouping tasks is derived from observation that tasks having near global optima will positively influence each other when there are genetic information transfers among them. When solutions of a task move further away from other tasks, the genetic information transfers between the task and others increase redundancy gradually. We call that the tasks with near global optima are 'similar' for simplicity. Despite that combining dissimilar tasks will increase diversity from the very beginning, convergence speed afterwards and the final results of these tasks within limited rounds leave much to be desired.

Similar tasks are grouped and dissimilar tasks are separated in the grouping operation. Mating between individuals is only allowed in the same group so that negative genetic transfers could be eliminated. Grouping tasks can be implemented in two levels: genotype and phenotype.

At the genotype level, tasks are grouped by genotypic similarity of their best individuals. We resort to the clustering technique like the bisecting K-means [Steinbach *et al.*, 2000] based on the *Manhattan* distance between individuals. Using one group for a searching area, M representatives are chosen from the best individuals to represent M groups or M search spaces one-on-one.

At the phenotype level, representatives conduct tests on others' districts to see any better exploration positions. We run tests between representative pairs rather than best individual pairs, which reduces the needed number of tests. Only

Algorithm 1: Grouping Tasks

Input: BI (best individuals)
Output: Groups of Individuals GB

$Groups \leftarrow bisectingKmeans(BI)$.
Randomly choose representatives (RG) of groups while one representative is corresponding to one group.
$k \leftarrow \lceil M/2 \rceil$. // $|Groups| = M$.
$kRG \leftarrow$ Randomly choose k representatives from RG.
$t \leftarrow k$.
Generate an empty set ERG for archive.
for $i = 1$ **to** k **do**
    $tRG \leftarrow$ Randomly choose t representatives from RG.
    Change the skill factor of all representatives in tRG to τ_i while τ_i is the skill factor of i-th representative of kRG.
    $ERG \leftarrow ERG \cup tRG$.
end
Evaluate the representatives in ERG with respect to their skill factors only.
Obtain the new best individuals (NBI) according to ERG.
for $nbi_j \in NBI$ **do**
    if nbi_j *is fitter than* bi_j **then**
        Remove j from its original group.
        Add j into the group represented by nbi_j.
    end
end
Update GBI.

partial representatives are selected to run the tests on other selected representatives, and their skill factors are changed to the formers for additional evaluations and the best evaluation results are chosen for every test representative. The skill factor of an individual is the one task, on which the individual is most effective. The structure of groups and the best individuals are updated by the results. The task grouping process is described in Algorithm 1.

In order to illustrate the procedure of grouping, assume there are four tasks from T_1 to T_4 whose best individuals are bi_* respectively in Fig. 2. The solid points are for the best individuals while each solid circle is for the exploitation space of each task constructed by its individuals. After the preliminary grouping, T_1 and T_2 reside in group g_1, and T_3 and T_4 are in group g_2. Assuming that bi_1 and bi_4 are selected as representatives of g_1 and g_2 respectively. Then bi_1 conducts a test in bi_4, which means that bi_4 changes its skill factors to bi_1's. Next, bi_4 is evaluated in task T_1 and luckily achieves better results than bi_1's. It shows that adding task T_1 into group g_2 will improve solutions to T_1 due to a better evolution of bi_1. Finally two new groups: $\{T_2\}$ and $\{T_1, T_3, T_4\}$ are obtained.

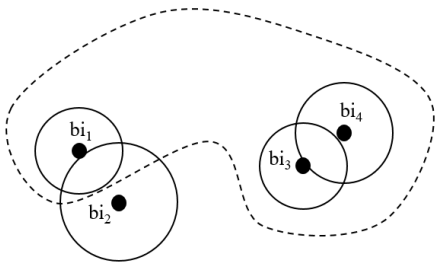


Figure 2: Grouping tasks.

3.2 Selection

Selection Criterion

The existing MFEA algorithm adopts a fitness-based selection criterion for effectively transferring elite genes between tasks and best individuals are always kept for each task, which implies a large accumulation of elite genes. However, population diversity is necessary when it becomes a bottleneck against the genetic information transfer. Meanwhile, the uncertainty of genetic information brought by other tasks can't ensure local diversity of a particular task, which is useful for an efficient search [López *et al.*, 2010; Ishibuchi *et al.*, 2008; Črepinšek *et al.*, 2013]. In GMFEA, we propose a new selection criterion keeping a balance between individual fitness and diversity in a population.

We redefine fitness scalar (*FS*) as a fitness estimator to adjust factorial cost of individuals evaluated for different tasks to a common scale.

Definition 1 (Fitness Scalar): For each individual $p_i \in P$ on task T_j, *FS* is computed in Eq. 1.

$$p_i.FS = (\Psi_j^i - lb_j)/(ub_j - lb_j) \tag{1}$$

where Ψ_j^i is the factorial cost of p_i on task T_j, lb_j and ub_j are minimum and maximum factorial costs of individuals on task T_j over all individuals in P.

We use crowding distance (*CD*) [Deb *et al.*, 2000; Deb *et al.*, 2002; Yang *et al.*, 2013] to approximately estimate individual diversity here.

Definition 2 (Crowding Distance): For each individual p_i, crowding distance is computed in Eq. 2.

$$p_i.CD = \sum\nolimits_{p_k \in N(p_i)} (1 - D(p_i, p_k)) \tag{2}$$

where $D(p_i, p_k) = \sum_{l=1}^{D} |p_i(l) - p_k(l)|$ measures the distance between p_i and p_j in the genotype level using a *Manhattan* distance, $p_i(l)$ denotes the l-th component of p_i as a vector form, and $N(p_i) = \{p_k | D(p_i, p_k) < neighbor\_size\}$ stands for neighbours of p_i ranged by the $neighbor\_size$.

Strictly speaking, CD is not a standard estimator of diversity, but density, as it serves for each individual instead of an entire population. However, maximising diversity of the population of individuals is a NP-hard problem and approximate algorithms are computationally expensive [Qu *et al.*, 2015; Nemhauser *et al.*, 1978]. We resort to a heuristic method of computing CD for each individual, which provides a good diversity of the population while using less computation.

Formally the selection criterion weighing against both fitness and diversity is defined in Eq. 3.

$$\min_i \{\alpha \times p_i.FS + (1 - \alpha) \times p_i.CD\} \tag{3}$$

where α is the balance factor.

Mating Selection

Mating selection aims at focusing search strength on the most promising individuals [Yang *et al.*, 2013]. We adopt it as a preparation for genetic information exchange. In this paper, we use a binary selection technique and consider both the fitness and diversity to form a promising mating pool for each group. Two individuals are randomly selected one of which

Algorithm 2: Mating Selection

Input: $Groups$ (grouped tasks), P (population), N (population size), α
Output: New populations NP

Reshuffle P.
Generate an empty set NP for archive.
for $i = 1$ **to** $N/2$ **do**
  $p_1 \leftarrow (2 \times i - 1)$-th individual of P.
  $p_2 \leftarrow (2 \times i)$-th individual of P.
  if $(\alpha \times p_1.FS + (1-\alpha) \times p_1.CD) < (\alpha \times p_2.FS + (1-\alpha) \times p_2.CD)$ **then**
    $NP \leftarrow NP \cup p_1$.
  else
    $NP \leftarrow NP \cup p_2$.
  end
end

Algorithm 3: Environmental Selection

Input: P (population), T (the set of tasks), $N_{\tau'}$ (the size of individuals required by task τ') `// τ' is also representing the relevant skill factor`
Output: New population NP

Generate an empty set NP for archive.
$\{p.CD | p \in P\} \leftarrow 0$.
for $\tau' \in T$ **do**
  $P' = \{p \in P | p.\tau = \tau'\}$.
  Generate a temporary set tP for archive.
  while $|tP| < N_\tau$ **do**
    $q \leftarrow \min_{p \in P'} \{\alpha \times p.FS + (1-\alpha) \times p.CD\}$.
    $tP \leftarrow tP \cup q$.
    $P' \leftarrow updateCD(P', q)$.
  end
  $NP \leftarrow NP \cup tP$.
end

is chosen to stay in the mating pool while the other is abandoned. Eventually only half of a population is allowed to be mated, which significantly reduces computational cost in every evaluation call. Details are summarised in Algorithm 2.

Environmental Selection

Environmental selection aims to maintain promising candidate individuals for a new generation. An intuitive thought is to design a selection strategy based on fitness. However, selecting similar or even identical genotypic individuals may lead to a significant loss of diversity. In GMFEA, we follow the criterion (Eq. 3) in the selection. First all individuals' crowding distances (CDs) are set as zeros. For each task, a certain number of individuals are selected using CDs. After an individual q has been picked out, we update the CD (in Algorithm 4) for q's remaining neighbours. Thus the selection probability of q's neighbours drops for keeping a large diversity of the population. The main procedure of environmental selection is summarised in Algorithm 3.

4 Experimental Study

All experiments are performed on a desktop with Intel i7 CPU (3.4 GHz) and 8GB RAM.

Algorithm 4: Update Crowding Distance

Input: P (population), q (selected individual), D (distance matrix), $N(q) = \{p \in P | D(p,q) < neighbor\_size\}$

for $p \in N(q)$ **do**
  $p.CD \leftarrow p.CD + 1 - D(p,q)$.
end
Normalize CD.

Table 1: Search spaces of functions

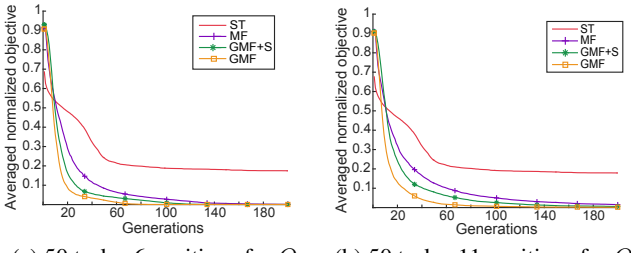
Function	Range	Function	Range
Griewank	[-600,600]	Rastrigin	[-5,5]
Ackley	[-32,32]	Weierstrass	[-0.5,0.5]

4.1 Function Optimisation

We begin the cross-domain experiments with four benchmark functions for continuous optimisation that are commonly used in the closely related literature [Yew-Soon Ong *et al.*, 2006; Chauhan *et al.*, 2013; Nguyen *et al.*, 2007; Le *et al.*, 2009]: a) *Griewank* function: $1 + \sum_{i=1}^{n} \frac{z_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{z_i}{\sqrt{i}})$; b) *Ackley* function: $20 + e - 20\exp(-0.2((\sum_{i=1}^{D} z_i^2)/D)^{(1/2)}) - \exp(\sum_{i=1}^{D} \cos(2\pi z_i)/D)$; c) *Rastrigin* function: $\sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10)$; and d) *Weierstrass* function: $\sum_{i=1}^{D}(\sum_{k=0}^{20}(0.5^k \cos(2\pi \cdot 3^k(z_i + 0.5)))) - D\sum_{k=0}^{20}(0.5^k \cos(2\pi \cdot 3^k \cdot 0.5))$.

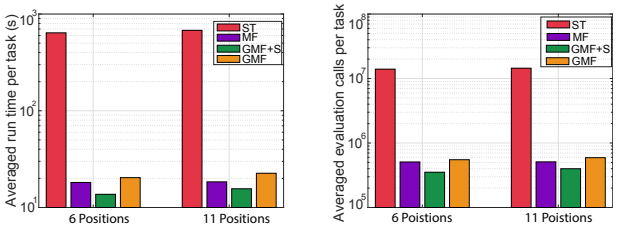
In above functions, $z = M_* \times (x - O_*)$ and M_* are rotation matrices which influence the landscapes and O_* decides global optima of each function. Their search spaces are summarized in Table 1. We perform experiments by varying combination of functions, M_* and O_*. One task is instantiated after one function, which represents task's type, is chosen, and its M and O are randomly generated. D is set to 30, so the search dimension is 30.

We present two sets of cross-domain experimental results. One is in Figs. 3 - 5 that compare the performance of a single task EA (denoted as ST) to the MFEA variants. MF denotes MFEA, GMF denotes MFEA plus task grouping strategy, and GMF+S denotes GMF integrated with the new selection criterion. It is noted that the mating selection is introduced along with the new selection criterion in GMF+S and largely reduces the evaluation calls by nearly a half. 50 tasks and a population of 250 individuals evolve over 200 generations. All the results are averaged over 20 independent runs.



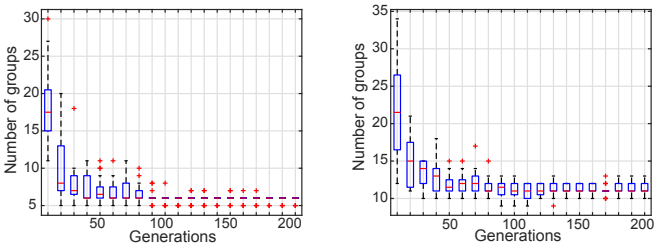
(a) 50 tasks, 6 positions for O (b) 50 tasks, 11 positions for O

Figure 3: Convergence trends of 50 tasks



(a) Averaged run time (b) Averaged evaluation calls

Figure 4: Running time and evaluation calls of 50 tasks.



(a) 50 tasks, 6 positions (b) 50 tasks, 11 positions

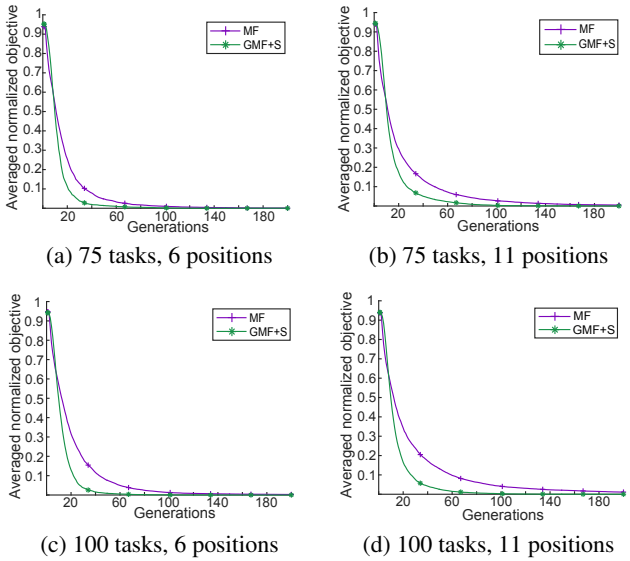
Figure 5: The changes of number of groups (median, minimum and maximum numbers) of GMF+S

The random mating probability (rmp) of MFEA keeps the same parameter value 0.3 as in the original paper [Gupta *et al.*, 2016]. GMFEA variants (GMF and GMF+S) have a full crossover and a mutation probability of 0.8 among members in the same group, and the normalised $neighbour\_size$ value is set to 0.4. Due to the quasi-Newton method in MFEA variants that highly strengthens the local search, we resort to a relatively big $\alpha = 0.95$ for a relatively small demand of a population diversity in the experiments. The value of α is decided by a trade-off between the run time and the objective values in the experiments (which is not shown due to the limited space).

Fig. 3 shows the convergence trends of 50 tasks with different position sets. The averaged normalised objective value is computed by $1/(20 \times 50) \times \sum_j \tilde{f}_j$, where $\tilde{f}_j = (f_j - (f_j)_{min})/((f_j)_{max} - (f_j)_{min})$, j represents task T_j, $(f_j)_{min}$ and $(f_j)_{max}$ are the minimum and maximum objective values among all runs of ST and MFEA variants corresponding to task T_j. 250 individuals serving for a single task earns the best beginning in ST. However, all the MFEA variants defeat ST after several generations because ST is quickly trapped in a local optimum. GMFEA variants exhibit a better convergence compared to MF, while the performance of GMF+S is between GMF and MF. For both cases, the variances are rather small (can't be shown in the figures), which implies a fast convergence and is consistent with the trends in Fig. 3.

In Fig. 4, MFEA variants show great time advantage over ST for running multiple tasks with the same size of population. GMF is more time-consuming than MF due to the time spent on grouping tasks and updating CDs. GMF+S is the fastest among all four algorithms as it reduces the number of individual evaluations.

Fig. 5 shows the changes of the number of groups of GMF+S. Global optima of 50 tasks distributed among 6 positions



(a) 75 tasks, 6 positions (b) 75 tasks, 11 positions

(c) 100 tasks, 6 positions (d) 100 tasks, 11 positions

Figure 6: Convergence trends of larger task sets

and 11 positions become stable within 100 generations, which endorses the reasonable strategy of grouping tasks and its utility. Fig. 6 focuses on the scalability tests of MF and GMF+S. Two larger task sets of 75 tasks and 100 tasks are considered in two position sets. The population size increases to 400 and other parameters keep the same. GMF+S outperforms MF and achieves larger scalability in all the cases.

4.2 Sudoku Puzzles

Sudoku is a puzzle game designed for a player to fill in a 9 x 9 grid with digits from 1 to 9. We draw part of Sudoku puzzles from the website (websudoku.com) while the rest of puzzles are hand-crafted.

The chosen Sudoku puzzles serve the purpose of providing intuitive benchmarks where although there may exist little similarity between tasks on the surface, there indeed exist strong underlying (latent) correlations between the optimum solutions of the tasks. It is contended that such examples are common in diverse real-world settings, thereby providing immense scope and motivation for multitask problem-solving.

Two sets of 30 Sudoku puzzles are considered in which 25 puzzles are manually generated while the rest are automated by our puzzle programs. One contains 15 groups of similar tasks while the other contains 10 groups of similar tasks. Fig. 7 gives examples of two pairs of similar tasks: A1-A2 and A3-A4. A population of 500 individuals evolve over 100 generations. GMF+S ($\alpha = 1/0.7$) denotes GMF+S with two α settings of 1 and 0.7. Values of other parameters are the same as those in the previous domains.

We compute the population diversity through the well-known entropy measurement [Tsujimura and Gen, 1998; Gupta and Ong, 2016]. $E_{(r,c)}$ denotes the diversity at each cells and is computed as follows:

$$E_{(r,c)} = -\sum_{j=1}^{9} pr(allele_{(r,c)} = j) \cdot \log_9(pr(allele_{(r,c)} = j)),$$

where r and c are the row and column indexes, $allele_{(r,c)}$ denotes the digit filled in the cells (r, c), and $pr(x)$ represents the occurrence probability of 'x' in one generation.

9	1		7					
	3	2	6		9		8	
		7		8		9		
	8	6		3		1	7	
3								6
	5	1		2		8	4	
		9		5		3		
	2		3		1	4	9	
					2		6	1

Sudoku: A1

9	1			4	5			
							8	
		7	2	8				5
2	8						7	
3	9			7	8			6
7		1		2			4	3
1				5	7	3		
			3		1			7
	7	3		9	2	5		

Sudoku: A2

6			7		5	4		3
		3		2		7		
					6		5	1
2		5		6		9		4
	7						1	
9		1		7		5		8
5	3		8					
		8		9		3		
7		6	1		3			9

Sudoku: A3

	9	2			5	4		3
		3	4					6
8				3				
		5	3		1			4
3				8		6		2
9		1	2		4			
		9		4		1		7
	1		6				2	5
7	2			5		8		

Sudoku: A4

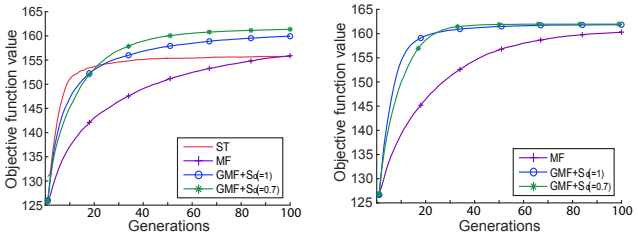
Figure 7: Examples of Sudoku puzzles. A1 and A2 are high similar puzzle pairs so are the pairs of A3 and A4. Other puzzle pairs composing by the four puzzles are of low similarity.

Subsequently, the population entropy is the sum of diversity of cells: $E = \frac{1}{81}\sum_{r=1}^{9}\sum_{c=1}^{9} E_{(r,c)}$.

Fig. 8 depicts the convergence trends of two sets of 30 Sudoku puzzles. We don't add the performance of ST in Fig. 8 (b) since it has similar trends in Fig. 8 (a). As shown in the first set of experiments (in Fig. 4), GMF is more time-consuming than GMF+S while achieving similar objectives. Hence we don't show GMF in the second set. We roughly divide Fig. 8 (a) into two stages by the $20th$ generation and divide Fig. 8 (b) by $30th$ generation. In both the figures, ST converges in the fastest manner, followed by GMF+S ($\alpha = 1$), GMF+S ($\alpha = 0.7$) again, and the slowest MF in the former stage. In the latter stage, GMF+S ($\alpha = 0.7$) converges first, followed by GMF+S ($\alpha = 1$), MF and ST again.

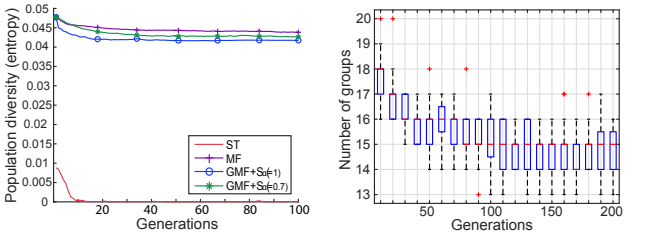
We account for the convergence performances accompanied by population diversities in Fig. 9 (a). Only the population entropy of 15 groups of similar tasks case is given because the other is similar. ST has the lowest population diversity, which helps its fast convergence in the former stage; however, it is easily trapped in a local optimum. Although ST offers each task the whole population of individuals, GMF+S ($\alpha = 0.7$) has a population diversity between MF and GMF+S ($\alpha = 1$) and finally convergences to the best against all the other algorithms. On the one hand, GMF+S ($\alpha = 0.7$) keeps a higher population diversity than GMF+S ($\alpha = 1$), which indeed accelerates the convergence speed in the latter stage. On the other hand, MF has the highest population diversity but performs worse than GMF+S, which shows that the unlimited population diversity derived from too many negative information transfers is harmful to the convergence. Fig. 9 (b) shows the changes of number of groups of GMF+S ($\alpha = 0.7$). After 50 generations, the number of groups keeps stable around 15.

In addition, we further investigate individual similarity in ST. In every iteration, we sample 300 individuals for each task (A1-A4) in terms of individual fitness in a population,



(a) 15 groups of similar tasks (b) 10 groups of similar tasks

Figure 8: Convergence trends of 30 Sudoku puzzles



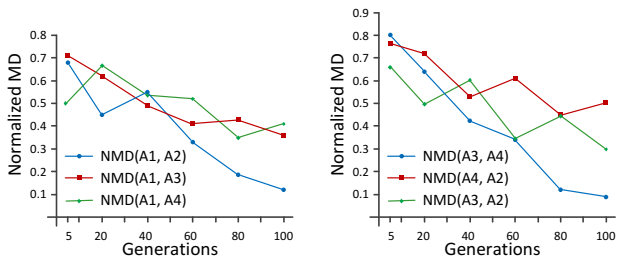
(a) Population entropy (b) The changes of number of groups of GMF+S ($\alpha = 0.7$)

Figure 9: Other metrics of 15 groups of similar tasks

and calculate the normalized $Manhattan$ distance (MD) for every pair of individuals. Fig. 10 shows the individual similarities that are drawn from different pairs of tasks. In Fig. 10 (a), individuals from similar tasks A1 and A2 have low distance leading to good similarity. This is also expected in Fig. 10 (b) when individuals from task A3 and A4 are compared. The experimental results verify our hypothesis that tasks having near global optima are similar, which motivates the idea of grouping tasks in MFEA, and indicates the necessary of introducing diversity in the new selection process.

5 Conclusion and Future Work

We propose the group-based MFEA that is scalable to deal with the optimisation of a significant number of tasks that can't be solved by the plain MFEA. The grouping is carefully designed using clustering algorithms and the selection is based on two well-formulated components. The performance is demonstrated by different angles over multiple domains. The results suggest effective performance with good efficiency achieved by GMFEA, which leads to promising solutions to a general many-task solver in many applications like a cloud platform. We notice that the algorithm cannot handle well the situation that few similar tasks exist in the task set, e.g. in an extreme circumstance that each group contains only one task. We will deal with this situation in the future work.



(a) Similarity between A1 and A2(b) Similarity between A3 and A4

Figure 10: Individual similarity for different sets of tasks

References

[Bäck *et al.*, 1997] T Bäck, U Hammel, and H.-P. Schwefel. Evolutionary Computation: Comments on the History and Current State. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[Caruana, 1997] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997.

[Chauhan *et al.*, 2013] Pinkey Chauhan, Kusum Deep, and Millie Pant. Novel inertia weight strategies for particle swarm optimization. *Memetic Computing*, 5(3):229–251, 2013.

[Cheng *et al.*, 2017] Mei-Ying Cheng, Abhishek Gupta, Yew-Soon Ong, and Zhi-Wei Ni. Coevolutionary multitasking for concurrent global optimization: With case studies in complex engineering design. *Engineering Applications of Artificial Intelligence*, 64:13–24, 2017.

[Coello, 2006] CA Coello Coello. Evolutionary multiobjective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, 2006.

[Črepinšek *et al.*, 2013] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and Exploitation in Evolutionary Algorithms. *ACM Computing Surveys*, 45(3):1–33, 2013.

[Deb *et al.*, 2000] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Parallel Problem Solving from Nature PPSN VI*, pages 849–858, 2000.

[Deb *et al.*, 2002] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[Fonseca and Fleming, 1995] Carlos M Fonseca and Peter J Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[Guo and Yang, 2015] Shu Mei Guo and Chin Chang Yang. Enhancing differential evolution utilizing eigenvector-based crossover operator. *IEEE Transactions on Evolutionary Computation*, 19(1):31–49, 2015.

[Gupta and Ong, 2016] Abhishek Gupta and Yew-soon Ong. Genetic Transfer or Population Diversification? Deciphering the Secret Ingredients of Evolutionary Multitask Optimization. jul 2016.

[Gupta *et al.*, 2016] Abhishek Gupta, Yew Soon Ong, and Liang Feng. Multifactorial Evolution: Toward Evolutionary Multitasking. *IEEE Transactions on Evolutionary Computation*, 20(3):343–357, 2016.

[Ishibuchi *et al.*, 2008] Hisao Ishibuchi, Kaname Narukawa, Noritaka Tsukamoto, and Yusuke Nojima. An empirical study on similarity-based mating for evolutionary multiobjective combinatorial optimization. *European Journal of Operational Research*, 188(1):57–75, 2008.

[Le *et al.*, 2009] Minh Nghia Le, Yew Soon Ong, Yaochu Jin, and Bernhard Sendhoff. Lamarckian memetic algorithms: Local optimum and connectivity structure analysis. *Memetic Computing*, 1(3):175–190, 2009.

[López *et al.*, 2010] Edgar Galván López, James Mcdermott, Michael O Neill, and Anthony Brabazon. Towards an Understanding of Locality in Genetic Programming. *Genetic And Evolutionary Computation Conference*, pages 901–908, 2010.

[Nemhauser *et al.*, 1978] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, 14(1):265–294, 1978.

[Nguyen *et al.*, 2007] Q H Nguyen, Y S Ong, and N Krasnogor. A study on the design issues of Memetic Algorithm. *2007 Ieee Congress on Evolutionary Computation, Vols 1-10, Proceedings*, pages 2390–2397, 2007.

[Qu *et al.*, 2015] Guannan Qu, Dave Brown, and Na Li. Distributed Greedy Algorithm for Satellite Assignment Problem with Submodular Utility Function. *IFAC-PapersOnLine*, 48(22):258–263, 2015.

[Steinbach *et al.*, 2000] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. *Proceedings of KDD-2000 workshop on text mining*, pages 1–20, 2000.

[Trunfio, 2015] Giuseppe Trunfio. A cooperative coevolutionary differential evolution algorithm with adaptive subcomponents. *Procedia Computer Science*, 51:834–844, 2015.

[Tsujimura and Gen, 1998] Y. Tsujimura and M. Gen. Entropy-based genetic algorithm for solving TSP. *1998 Second International Conference. Knowledge-Based Intelligent Electronic Systems. Proceedings KES'98 (Cat. No.98EX111)*, 2(April):21–23, 1998.

[Wright *et al.*, 2003] Alden H. Wright, Michael D. Vose, and Jonath E. Rowe. Implicit parallelism. In *Genetic and Evolutionary Computation*, pages 1505–1517, 2003.

[Yang *et al.*, 2013] Shengxiang Yang, Miqing Li, Xiaohui Liu, and Jinhua Zheng. A grid-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 17(5):721–736, 2013.

[Yew-Soon Ong *et al.*, 2006] Yew-Soon Ong, Zongzhao Zhou, and Dudy Lim. Curse and Blessing of Uncertainty in Evolutionary Algorithm Using Approximation. *2006 IEEE International Conference on Evolutionary Computation*, 639798:2928–2935, 2006.

[Zhou *et al.*, 2016] Lei Zhou, Liang Feng, Jinghui Zhong, Yew-Soon Ong, Zexuan Zhu, and Edwin Hsing-Mean Sha. Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem. In *IEEE Symposium Series on Computational Intelligence*, pages 1–8, 2016.