UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Jean Nunes Ribeiro Araújo

DIVERSITY-DRIVEN MIGRATION STRATEGY FOR DISTRIBUTED EVOLUTIONARY ALGORITHMS APPLIED TO LARGE-SCALE OPTIMIZATION PROBLEMS

Belo Horizonte
2023

Jean Nunes Ribeiro Araújo

DIVERSITY-DRIVEN MIGRATION STRATEGY FOR DISTRIBUTED EVOLUTIONARY ALGORITHMS APPLIED TO LARGE-SCALE OPTIMIZATION PROBLEMS

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Lucas de Souza Batista

Belo Horizonte
2023

Araújo, Jean Nunes Ribeiro.
A663d Diversity-driven migration strategy for distributed evolutionary algorithms applied to large-scale optimization problems [recurso letrônico] / Jean Nunes Ribeiro Araújo. - 2023.
1 recurso online (111 f. : il., color.) : pdf.

Orientador: Lucas de Souza Batista.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 93-104.

Bibliografia: f. 105-111.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Algoritmos - Teses. 3. Modelos matemáticos - Teses. 4. Otimização - Teses. I. Batista, Lucas de Souza. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pela bibliotecária Ângela Cristina Silva - CRB-6/2361
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG.

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"DIVERSITY-DRIVEN MIGRATION STRATEGY FOR DISTRIBUTED EVOLUTIONARY ALGORITHMS APPLIED TO LARGE-SCALE OPTIMIZATION PROBLEMS"

JEAN NUNES RIBEIRO ARAUJO

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica. Aprovada em 17 de agosto de 2023. Por:

Prof. Dr. Lucas de Souza Batista
DEE (UFMG) - Orientador

Prof. Dr. Cristiano Leite de Castro
DEE (UFMG)

Profa. Dra. Elizabeth Fialho Wanner
Computer Science (Aston University)

Prof. Dr. Rodrigo César Pedrosa Silva
Departamento de Computação (UFOP)

Prof. Dr. André Luiz Maravilha Silva
Departamento de Informática, Gestão e Design (CEFET-MG)

Documento assinado eletronicamente por **Lucas de Souza Batista**, **Professor do Magistério Superior**, em 17/08/2023, às 18:06, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **André Luiz Maravilha Silva**, **Usuário Externo**, em 18/08/2023, às 14:29, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Cristiano Leite de Castro**, **Professor do Magistério Superior**, em 18/08/2023, às 15:03, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Elizabeth Fialho Wanner**, **Usuária Externa**, em 22/08/2023, às 10:53, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Rodrigo Cesar Pedrosa Silva**, **Usuário Externo**, em 22/08/2023, às 14:16, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2547339** e o código CRC **2C990AE7**.

Dedicado aos meus pais, irmão e sobrinhos.

“Os mais fortes de todos os guerreiros são estes dois: tempo e paciência.”

Leon Tolstói

Resumo

Problemas de otimização global em larga escala geralmente possuem milhares de variáveis de decisão e podem ser extremamente complicados de resolver usando metaheurísticas tradicionais. Para lidar com esses problemas, modelos distribuídos têm sido empregados com sucesso por muitos algoritmos evolucionários (AEs) na última década. Esses modelos fornecem meios para permitir a colaboração entre ilhas (subpopulações), permitindo assim projetar estratégias para lidar com a convergência prematura e a perda de diversidade. Por meio de migrações periódicas, muitos algoritmos evolucionários distribuídos (AEDs) têm sido propostos para melhorar o equilíbrio entre exploração global e local. Neste trabalho, nós apresentamos um mecanismo de migração baseado em diversidade, em que o momento da migração é determinado avaliando a perda de diversidade das ilhas. Chamamos essa estratégia de Estratégia de Migração Orientada à Diversidade (DDMS). Tendo os problemas de otimização global de larga escala como foco, construímos o DDMS dentro de um modelo Cooperativo Coevolutivo (CC) e usamos o *DE*/best/1 e o *SHADE* como otimizadores. Nós testamos o DDMS enviando o melhor indivíduo e chamamos essa estratégia de DDMS−BEST. Para competir com o DDMS−BEST, nós criamos uma estratégia para tentar garantir que o indivíduo migrante seja capaz de gerar uma diversidade que ajude uma determinada ilha a explorar novas regiões sem prejudicar sua saúde. Para isso, nós usamos um algoritmo de agrupamento *online* chamado *TEDA*-Cloud para gerar nuvens de indivíduos com boa aptidão que foram migrados anteriormente. Nesta estratégia, o indivíduo a ser migrado deve ser extraído de uma nuvem cuja distribuição populacional seja suficientemente diferente da distribuição populacional da ilha que solicitou a migração. Nós chamamos essa estratégia de DDMS−TEDA. Usando o conjunto de testes de otimização em larga escala CEC'2013 com 1000 variáveis de decisão, nós comparamos as estratégias que usam o DDMS com estratégias de migração tradicionais, ou seja, migrações com intervalo fixo e probabilístico. Experimentos computacionais em diferentes cenários mostraram que a incorporação da estratégia DDMS em um algoritmo cooperativo coevolutivo evolucionário distribuído levou a melhores resultados. Considerando os valores médios de erro, mostramos que tanto o DDMS−BEST quanto o DDMS−TEDA são melhores na grande maioria das funções e cenários testados. Em relação à diversidade, mostramos que o DDMS−TEDA obtem melhores resultados em 100% das funções testadas. No Apêndice A deste texto, destacamos também os resultados promissores do DDMS−TEDA em cenários com 50 e 100 variáveis.

Palavras-chave: problemas de otimização em larga escala; algoritmos evolucionários distribuídos; diversidade em algoritmos evolucionários; políticas de migração em algoritmos distribuídos.

Abstract

Large-Scale Global Optimization (LSGO) Problems usually have thousands of decision variables and can be extremely complicated to solve using traditional metaheuristics. To deal with these problems, distributed models have been successfully employed by many Evolutionary Algorithms (EAs) over the past decade. These models provide means to enable collaboration between multiple islands (subpopulations), thus allowing to design strategies to deal with premature convergence and loss of diversity. Through introducing periodic migrations, many Distributed Evolutionary Algorithms (DEAs) have been proposed to improve the balance between exploration and exploitation. In this work, we present a diversity-based migration mechanism, in which the moment to perform the migrations is determined by assessing the loss of diversity of the islands. We call this strategy Diversity-driven Migration Strategy (DDMS). Focusing on large-scale global optimization problems, we built DDMS into a Cooperative Co-evolutionary (CC) model and used *DE*/best/1 and *SHADE* as optimizers. We test DDMS by sending the best individual and call it DDMS−BEST. To compete with the DDMS−BEST, we create a strategy to try to ensure that the migrant individual is capable of generating a diversity that helps a given island to explore new regions without harming its health. For that, we use an online clustering algorithm called *TEDA*-Cloud to generate clouds of good fitness individuals that have been previously migrated. In this strategy, the individual to be migrated must be extracted from a cloud whose population distribution is sufficiently different from the population distribution of the requesting island. We call it DDMS−TEDA. Using the CEC'2013 large-scale optimization test suite with 1000 decision variables, we compare DDMS against traditional migration strategies, namely, fixed and probabilistic interval migrations. Computational experiments with different scenarios showed that incorporating the DDMS strategy in a Cooperative Co-evolution Distributed Evolutionary Algorithm (CCDEA) led to better results. Considering the average error values, we show that both DDMS−BEST and DDMS−TEDA are better in the vast majority of functions and scenarios tested. Regarding the diversity, we showed that DDMS−TEDA gets better results in 100% of the functions. In Appendix A of this text, we also highlight the promising results of the DDMS−TEDA in scenarios with 50 and 100 variables.

Keywords: large-scale optimization problems, distributed evolutionary algorithms, diversity in evolutionary algorithms, migration policies in distributed evolutionary algorithms.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Presentation

In the last years, several Evolutionary Algorithms (EAs) have emerged as efficient metaheuristics to solve complex, multidimensional, non-differentiable, and multimodal optimization problems. In spite of their prominent ability, many studies have shown the difficulties of EAs in dealing with Large-Scale Global Optimization (LSGO) problems, which usually have more than 100 decision variables. In fact, the curse of dimensionality is a problem that degrades the performance of the EAs mainly because the complexity of the optimization problem grows exponentially as the number of variables increases (Li et al., 2013, Chen et al., 2015). In these cases, it is a great challenge to explore the entire search space efficiently during the evolution process when the decision space is extremely wide (Ali et al., 2015, Sudholt, 2020, Jian et al., 2020).

To address this issue, distributed models have been utilized and EAs have shown to be easy to adapt to run in these environments (Gong et al., 2015). A common distributed policy is to break a population into small subpopulations. From there, each subpopulation evolves within an independent island. So, it is more likely to explore different regions of the search space. Many Distributed Evolutionary Algorithms (DEAs) use a communication policy that involves migrating individuals between islands periodically. This exchange of information helps to promote both convergence to promising regions and population diversity (Gong et al., 2015).

The literature has demonstrated that DEAs tend to outperform sequential EAs (Lorion et al., 2009, Ishimizu and Tagawa, 2010, Zhang et al., 2013, Apolloni et al., 2014, Gong et al., 2015). In fact, the benefit of having islands exploring multiple regions of the search space tends to outweigh the cost of performing periodic migrations (Gong et al., 2015, Lynn et al., 2018, Zhan et al., 2022). However, the performance of DEAs depends on some parameters. Two of the main

parameters are migration frequency and policy. The migratory frequency concerns defining when to migrate. Generally, higher migration frequencies can promote faster convergence but the algorithm may get trapped. On the other hand, when the migration frequency is lower, the global exploration ability tends to be higher but the algorithm can converge much slower. This trade-off between exploration and exploitation demonstrates that the choice of migratory frequency is essential to achieve good performance and usually such a parameter is problem-dependent. However, no work has considered a proper estimation of the instant to perform the migration. On the contrary, many works prefer to carry out migrations at fixed or probabilistic intervals (Gong et al., 2015, Sudholt, 2015, Ge et al., 2017, Abdelhafez et al., 2019, Duarte et al., 2021, Li and Gonsalves, 2022).

The migratory policy concerns choosing which individual will be sent by the sending island[1] and which individual will be replaced on the requesting island[2]. The most usual policy is to send the best individual, which replaces a random individual on the requesting island. However, such an approach can cause an increase in selective pressure since the subpopulations can become very similar over the generations. This occurs because the best individuals will be spread between the islands through subsequent migrations, which eventually will reduce the diversity. The side effect of this approach is that the algorithm tends to converge to a local-optimal solution. A more adequate approach involves migrating individuals that are not too similar to the requesting island population and, at the same time, these migrated individuals should have good fitness values. However, most works do not explore this possibility (Gong et al., 2015, Sudholt, 2015, Ge et al., 2017, Abdelhafez et al., 2019, Li and Gonsalves, 2022).

In short, although distributed models are designed to prevent the premature convergence problem and improve diversity, these models are subject to fall into the same problems as sequential EAs when frequency and policy of migrations are defined empirically or even defined without any criteria, which usually implies the occurrence of out-of-time, unnecessary and irrelevant migrations (Gong et al., 2015, Sudholt, 2015, 2020). This work explores these issues by proposing an approach to estimate the instant to perform the migrations based on the loss of diversity of the islands. In addition, a diversity-driven migration strategy is proposed with the aim of promoting a *productive diversity*[3] on the requesting island.

[1] **Sending island:** it receives a request from a requesting island and sends a selected individual to it.

[2] **Requesting island:** it requests a migration and receives an individual from the sending island.

[3] The concept of *productive diversity* is derived of the idea presented in McGinley et al. (2011), which defines the *healthy diversity* term as a diverse population of highly-fit (healthy) individuals, capable of adapting quickly to fitness landscape change and well-suited to the efficient optimization of multimodal fitness landscapes.

1.2 Motivation

To our knowledge, there are no previous works in the literature that consider, in the same proposal, the need to properly choose the instant of migration and the individual to be migrated when attempting to develop distributed evolutionary algorithms. Ignoring these factors can prevent evolutionary algorithms to take advantage of the full capabilities of distributed models in LSGO problems (nonlinear problems with hundreds or even thousands of real variables).

Given this gap, this work is motivated by the importance of: i) developing a proper approach to estimate the instant of migration considering that the loss of diversity on the islands can be asynchronous, and ii) defining the proper individual to be migrated taking into account the population distribution of the requesting island. In this way, we hope migrations end up having the desired effect, which is to help the requesting island to escape from local optima and explore promising new regions in LSGO problems.

1.3 Objectives

In a distributed environment in which multiple islands execute an evolutionary algorithm in parallel to solve an optimization problem, the main objective of this work is to propose a Diversity-driven Migration Strategy (`DDMS`) capable of promoting *productive diversity* applied to large-scale global optimization problems. We use the term *productive diversity* to say that a migration should generate timely diversity without severely compromising convergence (McGinley et al., 2011). To achieve this main objective, the specific objectives are defined:

- Assess recursively the loss of diversity considering that this can occur asynchronously.
- Design a distributed evolutionary algorithm where islands run subpopulations using a parallel implementation.
- Define rules that establish when a migration should occur between the islands, allowing it to be utilized into the implemented distributed evolutionary algorithm.
- Create a Cooperative Co-evolution Distributed Evolutionary Algorithm (CCDEA) that includes the proposed Diversity-driven Migration Strategy (`DDMS`) in order to solve Large-Scale Global Optimization (LSGO) problems.
- Develop a strategy based on clustering capable of increasing the positive effect generated by individuals migrated to the requesting island regarding diversity.

- Evaluate the proposed approaches to validate their efficacy and to compare them against the traditional approaches that do not consider diversity as a determining factor to specify when and how to carry out migrations.

1.4 Contributions

The main contributions of this work are:

1. The proposal of a proper approach to estimate recursively the instant to perform a migration in a distributed evolutionary algorithm considering stagnation and convergence of the population.
2. The proposal of specific strategies that determine how to migrate in a distributed evolutionary algorithm taking into account that the migrated individual must be an agent capable of generating diversity on the islands.
3. The proposal of a Cooperative Co-evolution Distributed Evolutionary Algorithm (CCDEA) that executes the proposed migration strategies to optimize large-scale continuous problems.

The contributions mentioned above have resulted in two publications in scientific journals. These publications are:

- **Araujo, J.N.R., Batista, L.S. & Monteiro, C.C.** Improving proactive routing with a multicriteria and adaptive framework in ad-hoc wireless networks. Wireless Netw 26, 4595–4614 (2020). https://doi.org/10.1007/s11276-020-02366-4
- **Araujo, J.N.R. & Batista, L.S.** A Diversity-driven Migration Strategy for Distributed Evolutionary Algorithms. Swarm and Evolutionary Computation (2023). https://doi.org/10.1016/j.swevo.2023.101361

In the first article, a differential evolution algorithm for multiobjective optimization is used to solve a node deployment problem in ad-hoc wireless networks. In this work, some limitations of the evolutionary algorithm are discussed when applied to problems with a high number of nodes (variables). As future work, a distributed evolutionary algorithm is suggested to solve problems with a large number of nodes and sensors.

In the second article, the cornerstone of the strategies proposed in this text are presented. Specifically, contributions 1 and 2 above are introduced under the name `DDMS` (Diversity-driven Migration Strategy). The strategy is tested in problems with 10, 30, 50, and 100 variables, showing very promising results when compared against traditional migration strategies.

1.5 Organization of the text

The chapters are briefly described below:

This Chapter 1 introduces the issues addressed in this work and presents the motivation, the objectives and contributions of this thesis.

Chapter 2 describes one of the main challenges of search algorithms, which is to design mechanisms capable of balancing exploration and exploitation of the search space. Population diversity is presented as a relevant index used to identify stagnation and premature convergence. Then, distributed evolutionary algorithms are introduced as promising tools to improve exploration and exploitation. The relevant parameters used to evaluate the performance of a distributed model are also presented, followed by a discussion of the main studies in the literature on distributed evolutionary algorithms.

Chapter 3 introduces the decomposition methods utilized to decompose a LSGO problem into several low-dimensional subproblems and presents the architecture of the proposed CCDEA (Cooperative Co-evolution Distributed Evolutionary Algorithm).

Chapter 4 presents the mechanism to identify the stagnation and convergence in each dimension. Three rules are established to determine when to migrate. The first rule determines that migration should occur when all dimensions converge or stagnate. The second rule establishes that migration must occur in a given generation with a small probability. A final rule is introduced so that the probability of migration increases as the algorithm evolves. Furthermore, the optimization algorithms utilized are described.

In Chapter 5, specific algorithms are presented for the implementation of the proposed diversity-driven migration strategy. Then, the framework for detecting outliers and the algorithm for performing clustering are detailed.

Chapter 6 describes the computational experiments performed to evaluate the proposed diversity-driven migration strategy using the suite of 15 benchmark functions for large-scale global optimization provided by CEC'2013 (Li et al., 2013). The gains in terms of convergence and diversity are assessed in detail when compared with traditional migration strategies.

Finally, in Chapter 7 the conclusions are presented, ending with some ideas of continuity that can be explored in future works.

In addition, Appendix A extends the analysis of the proposed strategy to problems with a moderate number of variables (50 and 100) using the suite of 30 benchmark functions for single-objective real-parameter numerical optimization provided by CEC'2014 (Liang et al., 2013).

Chapter 2

Distributed evolutionary algorithms to improve exploration and exploitation

2.1 Exploration and exploitation of search space

The main challenge of search algorithms is to explore vast and promising regions of the search space while generating good solutions by converging to a particular region. Countless applications have come across this question: how to exploit without losing the ability to explore and how to explore without losing the ability to exploit?

The authors in Črepinšek et al. (2013) describe that exploration is the process of visiting entirely new regions of a search space, while exploitation is the process of visiting those regions of a search space within the neighborhood of previously visited points. Generally, a metaheuristic is efficient when it is able to balance exploration and exploitation. However, this task is not trivial.

In essence, most current evolutionary algorithms have mechanisms to deal with exploration and exploitation. Many researches share the idea that the crossover/mutation operators are designed to explore, while exploitation is done by selection operators. However, this perception has changed as the area has developed.

- **Crossover operator:** This operator seeks to generate an offspring by mixing information from two or more parents. In general, such an operator is designed to combine good individuals in order to generate a new one even better than its parents. From this perspective, it can be mainly seen as an exploitation operator. However, a good crossover operator should also be able to generate explorers individuals (Črepinšek et al., 2013).
- **Selection operator:** Operator that seeks to drive the search toward the regions where the best individuals are located. From this point of view, selection is more about exploitation.

This is especially true if the selection pressure is high. On the other hand, the search tends to be more exploratory when the selection pressure is low. Therefore, it can be tuned to balance between exploration and exploitation (Črepinšek et al., 2013).

- **Mutation operator:** According to a given probability, this operator seeks to modify individuals randomly. Generally, this operation is applied to increase the structural diversity of a population. From this perspective, a mutation operator is more applied to exploration as it can recover the diversity lost during the selection phase. But, a mutation can also be seen as an exploitation operator because it preserves most of the individual's characteristics. Besides, the role of mutation can be very different depending on the algorithm used (Črepinšek et al., 2013).

In this context, the authors in Črepinšek et al. (2013) argue that it is difficult to predict if a crossover and/or mutation operator will generate exploiter or explorer individuals. In a nutshell, there is not a clear line that separates exploration and exploitation. In consequence, both operators can have both roles.

Achieving the balance has been the subject of numerous researches nowadays, mainly in the field of parameter setting. For example, if crossover and mutation rates are low, the search space can not be explored in depth. In such a case, the evolutionary algorithm and the hill-climbing algorithm will be alike. On the other hand, if crossover and mutation rates are very high, good solutions can be missed due to this exploratory feature. In such a case, the evolutionary algorithm will be closer to a random search (Karafotias et al., 2014, Eiben and Smith, 2015). It is important to highlight that these are general assumptions and, depending on the structure of the algorithm, may not apply in specific contexts. In fact, the parameters and operators set in EAs present many nuances and it is not always possible to define general rules (Hassanat et al., 2019, Drugan, 2019).

Besides that, researches in EAs have shown that control parameter setting is problem-dependent. A parameter setting can be optimal for a specific problem and might not be well adjusted for another problem (Smit and Eiben, 2009, Karafotias et al., 2014). In practice, different problems require different treatments in relation to exploration and exploitation. For instance, multimodal function optimization usually requires more exploration than unimodal function optimization.

Another important factor is the population size. It is a common belief that the larger the population size, the greater the search-space exploration tends to be. Generally, increasing the population size is a simple strategy to keep the population as diverse as possible. However, larger populations can also converge slowly and waste processing time. Besides that, some researches have shown situations in which the population size (small or large) has no significant effect on the quality of the final solution (Smit and Eiben, 2009, Črepinšek et al., 2013, Karafotias et al., 2014, Eiben and Smith, 2015).

Another issue is that the optimal value of the population size can change during the different stages of an evolution process. A classical idea is that a larger population is more needed in the early stages of the process. It is common to believe that EAs should start with exploration and then gradually change into exploitation. Such a policy can be easily applied to the mutation in which its rate decreases along with the evolution. The authors in Črepinšek et al. (2013) point out that such reasoning is generally correct, but this policy tends to face difficulties when solving multimodal problems with many optima or when dynamic environments are evolved. They argue that can occur premature takeover of exploitation over exploration.

2.2 Improving exploration and exploitation with distributed EAs

As discussed previously, evolutionary algorithms can suffer from premature convergence and loss of diversity in spite of their prominent ability. Distributed models have been utilized to lead with these issues. A standard policy is partitioning the population into small subsets as if they were islands. From there, each subset evolves independently with the aim of exploring different regions. Generally, such an autonomy helps to avoid premature convergence. In addition, a communication policy is implemented between the islands, which is called migration. The exchange of information among islands is a strategy to promote both convergence to promising regions and population diversity. An additional advantage of island-based algorithms is the possibility of running them on parallel/distributed hardware and, thus, reducing computational time. The authors in Lopes and de Freitas (2017), Wang et al. (2019) list some parameters that directly impact the island model performance in relation to the solution quality and the convergence speed. These parameters are summarized as follows:

- **Number of islands:** defines the number of subpopulations in the model;
- **Migration topology:** describes the communication schema between the islands. Figure 2.1 illustrates three commonly adopted topologies. In the ring, each island has two neighboring islands and they form a ring-shaped. The lattice is grid-shaped, and its neighboring islands are those with one difference at the location of either vertical or horizontal position. In the fully connected, each island is a neighbor of any other island;
- **Migratory rate:** defines how many individuals migrate from one island to another. The most common is to define that only one individual will be migrated, although there are several proposals that use more than one;
- **Synchronization type:** defines if the migration process is performed synchronously or asynchronously;

- **Migratory frequency:** defines the periodicity of migration process. It is quite common to define a fixed frequency or even to establish a probability of migration occurring. This parameter significantly influences the decreasing speed of population diversity;
- **Migratory policy:** describes which individuals will be copied and replaced when the migration process occurs. The commonly used policies are *best-to-worst*, *best-to-random*, and *random-to-random*, which stand for the best individual substituting to the worst individual, the best individual substituting to a random individual, and a random individual substituting to a random individual, respectively. This parameter significantly influences the process of restoring population diversity.

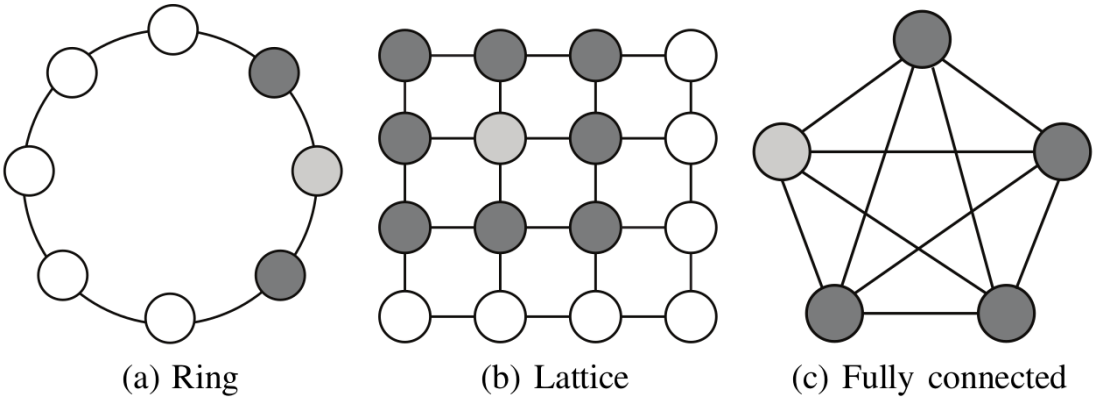


FIGURE 2.1: Migration topologies (Wang et al., 2019).

In this context, some prominent works in the literature are cited below with a special focus on migratory frequency and policy since these strategies can be used directly to maintain or recover diversity in island models or variants.

In Apolloni et al. (2008, 2014), the authors propose a simple distributed island-based DE in which the population is partitioned into two identical subpopulations. The subpopulations evolve independently with the same parameters and, after 100 generations, migration is made in order to promote diversity. In migration, one randomly selected individual is sent from one island to another. Incoming individual replaces randomly chosen local individual only if the former is better. This approach is simple, however, it is effective to improve the global search ability when compared to traditional DE. As said previously, traditional DE with a single population suffer from premature convergence problem when all individuals gather in the same valley. On the other hand, the migration made in fixed-interval policy has some drawbacks. For example, the work by Ishimizu and Tagawa (2010) demonstrates that can occur a trade-off between exploration and exploitation when different migration frequencies are used. The authors concluded that a higher communication frequency between the islands can speed up the convergence but the algorithm may get trapped. On the other hand, when the communication frequency is lower, the algorithm exhibits better global exploration ability but converges much

slower. They also show that the migration extent can bear a significant impact on performance. Therefore, choosing the migratory frequency is not a trivial task and usually, such a parameter is problem-dependent.

In Weber et al. (2009), the authors propose to group the subpopulations into two families. The first family has the role of exploring the decision space. The second family is composed of subpopulations with a population size that is progressively reduced. The idea is that the subpopulations belonging to the second family are highly exploitative. The solutions generated by the second family then migrate to the first family. The first family is composed of three subpopulations (islands) arranged according to a ring topology. Instead of fixed migration, a copy of the best individual of the subpopulation is sent to the next subpopulation in the ring with a given probability. Incoming individual replaces randomly chosen local individual, which is then discarded. The key idea is to divide the second family into two ages. In the initial age, two subpopulations have their population size progressively reduced during the optimization process until a determined number of fitness evaluations is achieved. During the second age, the two subpopulations evolve with the minimum population size and at each generation, the best individual is sent to one of the subpopulations of the first family. The main goal of the research is to promote exploration and exploitation synchronously. The progressive reduction of population size in the second family is derived from the common belief that EAs should start with exploration and then gradually change into exploitation. However, it can be difficult to demonstrate the effectiveness of separating islands for exploration and exploitation, since it is not possible to determine whether such strategies are actually producing exploratory or exploitative individuals. Besides, the migration is made without knowing if it is necessary to generate diversity or not.

Araujo and Merelo (2010) discuss the importance of defining a good migration policy. They argue the more effective way to cause the algorithm to converge significantly faster is to replace a random/worst individual with the best individual arriving from a home island. However, such an approach may lead the algorithm to a local-optimal solution as the populations become very similar after a few rounds of immigration. This occurs because the best immigrant individuals will be combined with other high-fitness individuals in the requesting island population, which eventually will reduce the diversity. On the other hand, the migration of a random individual can make the migration ineffective since the migrated individual tends to get lost in the evolutionary process of the new population. To overcome these drawbacks, the authors present a Genetic Algorithm (GA) that uses a diversity-driven approach that chooses the individuals to be sent to other islands based on the principle of *multiculturality*, that is, the individual sent should be different enough to the requesting island population. They demonstrate that such a policy outperforms the usual policy of sending the best or a random individual. This proposal indicates that migration also needs to be an adaptive process as far as possible. Another interesting discussion is about the synchronous/asynchronous communication between the islands. They argue that asynchronous communication does not have a negative effect on performance,

and it can even outperform synchronous one. However, the authors do not tackle the migration frequency problem and define it in a fixed way.

In Piotrowski et al. (2012), a new DE algorithm with separated groups is proposed. The main goal is to improve performance for difficult problems by distributing individuals into small subpopulations and defining diversified communication rules between them. The authors divide the population into some separate groups with 10 individuals in each group. They use two mutation and crossover strategies that differ significantly in terms of offspring selection. The first strategy, called *DE/rand/1/mod-either-or*, is well suited for exploration, but performs slowly in exploitation. On the other hand, the second strategy, called *DE/rand/1/exp*, does not perform well for exploring, but it speeds up exploitation and easily deals with separable functions. For each parent individual in an iteration, just one strategy is chosen randomly with equal probability. An interesting approach is introduced when the vectors of mutation are selected. Mostly, these vectors are randomly chosen from the small group. However, if the algorithm identifies the group has converged to a local minimum, the vectors are chosen from the whole population for a predefined number of iterations. This helps the group to get out from the local optimum. But, as information is spread quickly among members of small groups, using this single strategy can be inefficient. With this in mind, the authors also introduce a migration scheme based on the exchange of indices between individuals in a minimization problem. At the beginning of each iteration, the migration schema is triggered with probability *0.005*. For the first half of the population, if $f(\mathbf{x}_i) > f(\mathbf{x}_{i+1})$ then the individuals exchange the indices immediately. The idea is that the individuals with good and moderate fitness are scattered among different groups, but the ones with the poorest fitness are quickly moved into the single poorest group. For the second half of the population, the process is reversed; the individuals just exchange the indices if $f(\mathbf{x}_i) < f(\mathbf{x}_{i+1})$. In this case, the individuals with the best fitness are quickly moved to the elitist group. The combination of these strategies allows the algorithm to have various kinds of groups and, hence, adds diversity to the algorithm's search. As the authors define the migration frequency in a probabilistic way, the migration can occur after many generations with a group trapped in a local optimum. Besides that, the index exchange strategy can cause sudden changes in some groups and almost no change in other groups since the migration is not made directly between the groups.

In Meng et al. (2017), the authors use a GA method to implement a dynamic island model and a new migration schema based on similarity. The proposed island model starts with one island in the first generation. When the evolution reaches a specific generation, the spectral clustering algorithm is introduced to split the population into a set of new islands. Similar individuals are assigned to the same island. Then, each new island evolves independently. Every 50 generations, all individuals migrate to a pool wherein the spectral clustering algorithm is executed in order to reconstruct the islands according to the difference-based similarities of individuals. After a number of generations, the best individuals are considered potential global

optimized solutions. The authors evaluate the proposal on a set of combinatorial optimization problems including Job Shop Scheduler (JSSP), Travelling Salesman (TSP), and the Quadratic Multiple Knapsack (QMKP) problems. The performance was evaluated by fitness score and population diversity and compared with three migration schemes (ring, star-shape, and fully connected). They showed that the clustering model performs better than ring, star-shape, and fully-connected models because the clustering can effectively maintain diversity compared to the other ones. However, this similarity-based approach is extremely elitist as it reconstructs the clusters/islands entirely after in a fixed interval.

In Abdelhafez et al. (2019), the authors propose a distributed GA-based algorithm utilizing multiprocessors. They implement an algorithm in which the migration topology is based on the uni-directional ring topology. The authors argue that, different from more sophisticated topologies like lattice and fully connected topologies, the ring topology ensures local communications between subpopulations and smooth propagation of the information at a pace that does not make the algorithm converge too fast and keeps low the overall communication effort of the distributed GA. This is especially true because they use a model with 32 islands. Naturally, this requires more computational resources. In models with fewer islands, convergence tends to be faster because the propagation of the information will also be more accelerated. In their study, the migratory policy is *best-to-worst* and the migration is triggered every 5000 evaluations made in a subpopulation to let each island make sufficient exploration on its own before communications. They present an interesting result regarding the synchronization type. The asynchronous implementation is more time efficient than the synchronous implementation. The blocking communication of the synchronous implementation can cause delay since every island should wait to communicate with its neighbor island in the ring, leading to a waste of processing resources. On the other hand, the asynchronous algorithm uses unblocking communications, which means there are no idle cores waiting for communications. Besides that, asynchronous implementation is extremely useful when you want to design approaches in which the migration frequency is not fixed. In this case, the immigrant individuals merge into individuals in the requesting islands at any time, which can ensure a more productive diversity inside the islands.

In Zhang et al. (2013), the authors propose a new DE evolution in which the initial population is divided into subpopulations arranged by a grid topology. In this topology, subpopulations are denoted as nodes. Each node has four neighbors. Only those nodes that are close to each other (neighborhood) exchange information. Every 100 generations, the best individual in each node replaces the worst individual of its neighbor nodes. The study uses the Hooke-Jeeves algorithm, a local search method, mainly in the later evolution stage for enhancing the local search ability and improve the precision of solutions. Besides, two learning mechanisms to combine evolutionary search and local search heuristics are proposed. In short, the study finds out to prevent stagnation through a hybridization of DE with a local search algorithm and periodic migrations. However, nothing is done if the stagnation takes place as the migrations are fixed.

The authors in Ali et al. (2016) divide the population into two different tribes. Each tribe follows a different mutation strategy. The first strategy implements the *DE/current-to-pbest/1/bin* with a memory scheme to improve the parameter settings of F and CR. The second strategy, *DE/current-to-pbest/1/exp*, uses exponential crossover instead of binomial and adapts F and CR without using the memory scheme. Each of these mutation strategies utilizes different parameter settings to enhance diversity among the population. Each tribe has its own life cycle that controls its participation ratio for the next generation based on its recorded success. Besides, the population size is changed dynamically in each generation using a linear reduction method. However, the tribes are independent; there is no migration. With this strategy of dynamic adaptation of parameters in different tribes, the authors seek to improve *SHADE* algorithm (Tanabe and Fukunaga, 2013), which is an extension of *JADE (adaptive differential evolution with optional external archive)* algorithm (Zhang and Sanderson, 2009b). They demonstrate the algorithm is effective in solving unimodal, hybrid, composition functions, and most of the simple multimodal functions on the suite of 30 benchmark functions with 10D, 30D, 50D, and 100D from the IEEE CEC'2014 special session and competition on single objective optimization. Many studies have been dedicated to proposing adaptive strategies like this. The number of citations to the *JADE* (Zhang and Sanderson, 2009b) and *SHADE* (Tanabe and Fukunaga, 2013) papers in Google Scholar exceeds 2200 and 450, respectively, at the time of writing; not to mention papers that refer to other *JADE/SHADE*-based variants. In fact, these studies have shown that self-adaptive mechanisms can update the parameter settings to deal with different fitness landscapes (Awad et al., 2018). Piotrowski and Napiorkowski (2018) perform an inter-comparison among 22 different *JADE/SHADE*-based variants that were proposed between 2009 and 2017. They reveal that *SHADE*-based variants outperform those based on initial *JADE* and define the following chain of performance evolution: *DE* (Storn and Price, 1997) $\rightarrow$ *JADE* (Zhang and Sanderson, 2009b) $\rightarrow$ *JADE* with a weighted adaptation of control parameters (Peng et al., 2009) $\rightarrow$ *SHADE* (Tanabe and Fukunaga, 2013) $\rightarrow$ *L-SHADE* (Tanabe and Fukunaga, 2014), and from *L-SHADE* within the last years a number of successful variants were created (Ali et al., 2016). DE and its variants, like *L-SHADE*, have proven to be easy to adapt in distributed environments. Because of that, they have been gaining great prominence in specialized literature. We argue in this work that *L-SHADE* (Tanabe and Fukunaga, 2014), and possible variants (Ali et al., 2016), can become even more efficient when applied in a distributed environment with well-suited migrations. In addition, our proposal allows any EA can be employed as the optimizer.

In Duarte et al. (2021), the authors combine two strategies to preserve diversity in the island model. The first strategy uses different kinds of GAs for each island to force the generation of different individual characteristics and slow down the global diversity among the islands. The second strategy implements a novel migration policy that splits migrants into two classes: pursuer and avoider. An avoider has a task to promote diversity on the island, while a pursuer tries to enable exploration. A migrant of an island has a certain probability of being a pursuer or

avoider, depending on the island's current state. This mechanism seeks to ensure the individual migrates to the correct island dynamically, which will maintain diversity. The authors show that combining these strategies has great potential to preserve the overall island diversity.

Li and Gonsalves (2022) propose a new alternative to implement the island model, called the Stigmergy Island Model (Stgm-IM), inspired by the natural phenomenon of stigmergy. In this model, the population of each island is evolved by a distinct EA, which can imply different evolutionary characteristics applied in parallel to solve the problem. Each connection between the islands is weighted adaptively according to the attractiveness between each pair of islands. The idea is that the islands with the most suitable EAs to solve the problem become more attractive, and more solutions are directed to them.

In Price and Radaideh (2023), the authors propose an algorithm called AEO (Animorphic Ensemble Optimization), which assumes that a set of algorithms working as an ensemble can be a stronger performance across a wider range of optimization problems than any standalone algorithm. For this purpose, the algorithm is divided into two stages: the evolution phase and the migration phase. In the first phase of the AEO algorithm, a number of optimization strategies are specified to act as the ensemble for the optimization process. The authors use ES (Evolution strategy), PSO (Particle Swarm Optimization), and DE (Differential Evolution) as optimization strategies. In this phase, each population evolves apart from others as if they were on different islands. The migration phase consists of changing the island populations according to the performance of each algorithm. To this end, the authors propose strategies to remove and migrate individuals, as well as define the destination of individuals and the number of individuals to be removed and migrated.

These researches show the benefits of having a number of islands exploring multiple regions of the search space tend to outweigh the cost of performing periodic migrations. Another insight is that there is space in the literature to propose solutions that seek to carry out migrations only "when necessary", in addition to selecting migrated individuals in order to improve diversity without significantly compromising convergence.

2.3 Large-scale global optimization problems

Some researches have shown the difficulties of metaheuristics in dealing with Large-Scale Global Optimization (LSGO) problems, which usually have more than 100 decision variables (Li et al., 2013, Omidvar et al., 2021b, Zhan et al., 2022). In fact, even though the EA implements mechanisms to balance exploration and exploitation, the curse of dimensionality is a problem that degrades performance as the number of variables increases. This is because the complexity of the optimization problem grows exponentially as the dimensionality increases (Li et al., 2013,

Chen et al., 2015, Omidvar et al., 2021a,b). As a result, LSGO problems are also different when it comes to population size. Commonly, many algorithms define the number of individuals as a function of the number of variables. This can not be done for such problems, since it has an even greater impact on computational time, and can delay convergence prohibitively (Kazimipour et al., 2013, Chen et al., 2015, Omidvar et al., 2021a,b).

On the other hand, it is challenging to maintain population diversity during the evolution process with few individuals since the decision space is extremely wide. In LSGO problems, maintaining diversity is primordial to explore the entire search space efficiently (Ali et al., 2015, Sudholt, 2020, Jian et al., 2020). In this context, distributed evolutionary algorithms have attracted attention because of their capability of exploring multiple search regions by dividing the population into subgroups (Ge et al., 2017, Jia et al., 2018). Island-based models have shown very promisingly in many works (Ge et al., 2017, Meng et al., 2017, Wang et al., 2019).

The authors in De Falco et al. (2007) propose a locally connected topology, where each node is connected to 4 other nodes. The diversity policy determines each node must send a copy of its best individual to its neighbors every 5 generation. The algorithm is tested on larger-scale problems (500 and 1000-dimensional problems). This proposal generates more diversity since the migration is made routinely and the receiving island obtains individuals from more than one neighbor. However, this high and poorly targeted diversity can impair convergence since it is more prone to lose its ability to improve on its solutions. Also, the fixed-frequency migration might not be the better strategy as such a parameter tends to be problem-dependent.

Recent works have investigated strategies to solve LSGO problems in distributed environments. The work by Ali et al. (2015) divides the population into multiple islands. The idea employs different mutation strategies and control parameters for each island with the aim of driving the search toward promising regions. Some of these mutation strategies are adjusted to maintain a balance between exploration and exploitation. Each island evolves independently during a given interval to enhance diversity among the subpopulations. After this interval, migrations are performed. Some random individuals are selected from each island and they are sent to all other islands (fully-connected topology). The replacement is also done randomly and the 30% best performing individuals of each island can not be replaced. In short, the authors seek to maintain diversity by using different mutation strategies on each island and migrating random individuals. In general, such strategies are efficient. However, there is still the problem of defining when the migration should be made. That is, it is necessary to find a sufficient number of generations where the algorithm must evolve before mixing of information between the islands. The work does not address this issue and uses fixed-frequency migration.

In Ge et al. (2017), the authors propose to arrange the population dynamically. For that, they present two operators, namely mergence and split, which are executed by considering the contribution value of each island (subpopulation) to the entire evolution. The higher the contribution

value, the more the subpopulation has contributed to the optimization process. Otherwise, if the contribution value is lower, the subpopulation has been ineffective. In mergence operator, individuals that belong to ineffective subpopulations are merged into the good performing subpopulations. If a merged subpopulation becomes ineffective, the split operator selects randomly half of the individuals on the island and generates a new subpopulation to maintain diversity. Every 25 generations, a master node evaluates the contribution value of each island and determines if mergence or split should be executed. The migrations among islands are made in a probabilistic way. However, the migration frequency is empirically defined.

In Jia et al. (2018), the authors propose a two-layer distributed cooperative co-evolution architecture with adaptive computing resource allocation for large-scale optimization. The algorithm uses a decomposition procedure to divide a large-scale problem into subcomponents from the perspective of dimension. Then, each subcomponent initializes its own population and evolves using the Self-adaptive Differential Evolution with Neighborhood Search algorithm (SaNSDE) (Yang et al., 2008). During evolution, the subcomponents calculate their contribution to the global objective. After 20 generations, they send the best individuals and contributions to the master process to make the synchronization. Based on the contributions, the resource allocator reallocates the computing resources through an allocation algorithm that assigns more processors to the subcomponents which contribute more to the global objective. The key idea is to take limited computing resources from the subcomponents that contribute less and give them to the subcomponents that contribute more. In Zhang et al. (2019b), a similar framework is proposed. But, instead of punishing subcomponents, this study evaluates the contribution of each variable according to the historical information of the best overall fitness value, and each subcomponent is dynamically constructed based on these contribution information. The authors in Li et al. (2022) assume this idea to build an algorithm capable of performing adaptive resource allocation based on this contribution information. However, although these studies cover the possibility of dividing the population of a subcomponent into subpopulations, no migration scheme is used to try to maintain diversity.

2.4 Final discussion

Despite the advantages brought by the use of distributed metaheuristics to solve a wide variety of optimization problems, none of the studies reviewed in this research deal with the following four issues:

1. *Assess the loss of diversity in each dimension.* It is important to consider that an island can converge asynchronously in each dimension. Therefore, it can be useful to identify which dimensions need to be diversified. Some proposals assess whether the population

has converged. But, they diversify through migration schemes that fully replace one or more individuals (Meng et al., 2017, Ge et al., 2017).

2. *Determine when to migrate*. The migration should be triggered according to some measure that informs diversity has been lost. It is useful in trying to avoid unnecessary migrations, i.e., migrations that do not contribute to diversity because the subpopulation has not stagnated or do not contribute to convergence because the subpopulation is evolving. In addition, a timely migration is essential to design a subsequent strategy that adequately responds to the loss of diversity. Most studies focus on performing migrations at fixed or probabilistic intervals (Araujo and Merelo, 2010, Ishimizu and Tagawa, 2010, Piotrowski et al., 2012, Apolloni et al., 2014, Meng et al., 2017, Ge et al., 2017).

3. *Determinate how to migrate*. The migrated individual should be sufficiently different from the requesting island population in order to improve diversity. This helps the island to get out from the local optimum. However, it is important to ensure the migrated individual has good or moderate fitness so as not to harm convergence. Most studies do not carry out this analysis and focus on migrating the best individual or a random individual (De Falco et al., 2007, Ishimizu and Tagawa, 2010, Piotrowski et al., 2012, Apolloni et al., 2014, Ali et al., 2015, Meng et al., 2017).

4. *Implement a diversity-driven migration strategy for large-scale global optimization problems*. Recent works demonstrate there is an openness to propose diversity-driven migration strategies for large-scale problems. Generally, we use a decomposition procedure to divide a large-scale problem into subcomponents from the perspective of the dimension (Jia et al., 2018, Zhang et al., 2019b, Lopes et al., 2021, Li et al., 2022). In this way, the mechanism for assessing the loss of diversity must be adjusted to work within each subcomponent.

Table 2.1 summarizes the strategies used by these studies. This work proposes a diversity-driven migration strategy to improve the performance of a Distributed Evolutionary Algorithm (DEA) applied to Large-Scale Global Optimization (LSGO) problems. This analysis must consider that convergence/stagnation can occur asynchronously both in the islands and in the dimensions of the problem. This diversification is expected to be productive in the sense of helping the subpopulation to escape from local optimums and find promising new regions. The objective is that the evolution and the migrations are done by trying to balance exploration and exploitation. Then, we propose the following mechanisms to achieve such a *productive diversity*: i) migrations must be triggered according to some diversity loss rules; ii) migrated individuals must be sufficiently different from the requesting island population, and iii) migrated individuals must have the potential to drive the evolution towards promising new regions.

TABLE 2.1: Summary with features the referenced studies.

References	Migration policy		Communication	Handle asynchronous stagnation/convergence (Y/N)	Large-scale problems (Y/N)
	When	How			
Apolloni et al. (2008, 2014)	Fix	Random	Asynchronous	N	N
Ishimizu and Tagawa (2010)	Fix	Best	Synchronous	N	N
Weber et al. (2009)	Probabilistic	Best	Synchronous	N	Y
Araujo and Merelo (2010)	Fix	Multiculturality	Asynchronous	N	N
Piotrowski et al. (2012)	Probabilistic	Best and Worst	Synchronous	N	N
Zhang and Sanderson (2009b)	No migration	No migration	-	N	N
Tanabe and Fukunaga (2013)	No migration	No migration	-	N	N
Tanabe and Fukunaga (2014)	No migration	No migration	-	N	N
Ali et al. (2016)	No migration	No migration	-	N	N
Meng et al. (2017)	Fix	Similarity	Synchronous	N	Y
Abdelhafez et al. (2019)	Fix	Best	Synchronous and Asynchronous	N	N
Zhang et al. (2013)	Fix	Best	Synchronous	N	N
De Falco et al. (2007)	Fix	Best	Synchronous	N	Y
Ali et al. (2015)	Fix	Random	Synchronous	N	Y
Ge et al. (2017)	Probabilistic	Best	Synchronous	N	Y
Jia et al. (2018)	No migration	No migration	-	Y	Y
Zhang et al. (2019b)	No migration	No migration	-	Y	Y
Li et al. (2022)	No migration	No migration	-	Y	Y
Zhang and Sanderson (2009b)	No migration	No migration	-	Y	Y
Duarte et al. (2021)	Fix	Random selection	Synchronous	N	N
Li and Gonsalves (2022)	Fix	Best and Elitism Random	Synchronous	N	N
Price and Radaideh (2023)	Fix	Elistim	Synchronous	N	N

Chapter 3

Cooperative co-evolution distributed evolutionary algorithms for large-scale global optimization problems

3.1 General structure of a CCEA

To improve the performance of EAs for Large-Scale Global Optimization (LSGO) problems, a useful approach is to decompose the problem into several low-dimensional subproblems. In fact, solving lower-dimensional subproblems is faster than optimizing the complete problem due to the curse of dimensionality (Li et al., 2013, Chen et al., 2015). The algorithms that use these decomposition approaches are called Cooperative Co-evolution Evolutionary Algorithms (CCEAs) (Mahdavi et al., 2015, Mei et al., 2016, Cabrera, 2016). In CCEAs, the process to solve an optimization problem consists of two steps: 1) decomposition and 2) optimization, as shown in Figure 3.1.

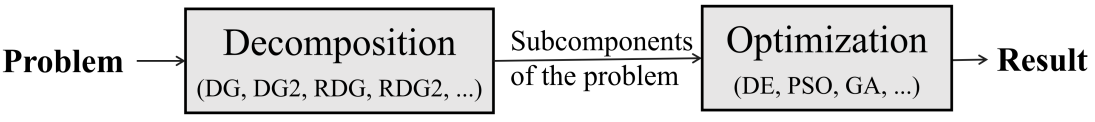


FIGURE 3.1: Structure of CCEA.

In the first step, CCEA decomposes a problem into several smaller subproblems. In the literature, these subproblems are also called subcomponents. Then each subproblem is associated with a separate optimizer with its own population. These two steps are relatively independent, which means one can choose any suitable decomposition method (e.g., DG, DG2, RDG, RDG2,

etc.) and any evolutionary algorithm (e.g., DE, PSO, GA, etc.) according to the real application (Yang et al., 2016, Lopes and de Freitas, 2017, Jia et al., 2018).

Figure 3.2 illustrates this process more closely for a serial CCEA where the optimization process uses a distributed model. This model is known as CCDEA (Cooperative Co-evolution Distributed Evolutionary Algorithm). While a subcomponent evolves, the others are held fixed. Thus, it is a serial architecture because there is only one subpopulation evolving. Within each subcomponent, a distributed model is executed according to the strategies proposed in the previous sections. In our implementation, we use 4 islands (subpopulations) per subcomponent.

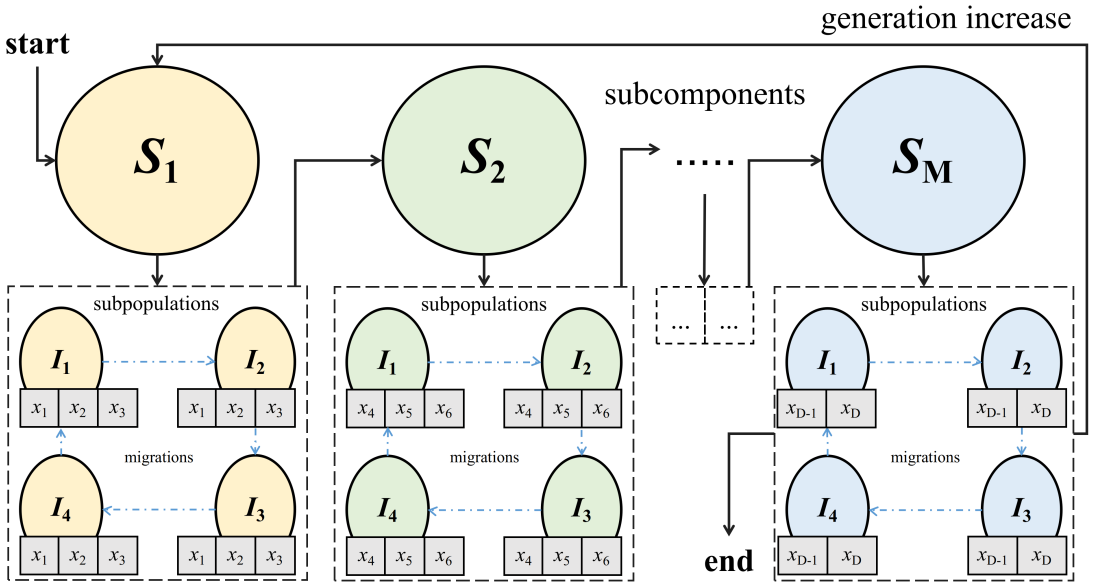


FIGURE 3.2: Optimization steps of a serial CCDEA.

3.2 Optimization separability

To execute the first step, the decomposition methods apply the concept of separability. An optimization problem is separable if it can be divided into two or more subproblems that can be solved independently. Thus, an optimization problem defined over the objective function, $f(\mathbf{x})$[1], is partially separable with M independent subcomponents if (Lopes et al., 2021)

$$\underset{\mathbf{x}\in\mathcal{F}}{\operatorname{argmin}} f(\mathbf{x}) = \left\langle \underset{\mathrm{s}_1\in\mathcal{F}_1}{\operatorname{argmin}} f(\mathrm{s}_1), \ldots, \underset{\mathrm{s}_M\in\mathcal{F}_M}{\operatorname{argmin}} f(\mathrm{s}_M) \right\rangle, \tag{3.1}$$

where M is the number of independent subcomponents, $\mathbf{x} = \langle x_1, \ldots, x_D \rangle \in \mathbb{R}^D$ is a decision vector of D dimensions, $\mathrm{s}_1 \in \mathbb{R}^{D_1}, \ldots, \mathrm{s}_M \in \mathbb{R}^{D_M}$ are disjoint sub-vectors of $\mathbf{x}$, $2 \leq M \leq D$,

[1] Variables in bold are vector variables.

and $D_1 + \ldots + D_M = D$. $\mathcal{F}_i \subset \mathbb{R}^{D_i}$ is the feasible set of the subproblem defined over s_i such that $\mathcal{F}_1, \ldots, \mathcal{F}_M$ are disjoint sets and $\mathcal{F}_1 \cup \ldots \cup \mathcal{F}_M = \mathcal{F}$ (Lopes et al., 2021).

A particular case of the definition above is when M is equal to D. In this case, the optimization problem is called fully separable, meaning that all decision variables can be searched independently.

A function can also have additive separability. An $f(\mathbf{x})$ is partially additively separable function if (Lopes et al., 2021):

$$f(\mathbf{x}) = \sum_{i=1}^{M} f\_sub_i(\mathrm{s}_i) \,, \tag{3.2}$$

where $\mathbf{x} = \langle x_1, \ldots, x_D \rangle \in \mathbb{R}^D$ is the global decision variable vector of D dimensions, s_i are disjoint sub-vectors from $\mathbf{x}$, and M is the number of independent subcomponents (Lopes et al., 2021). When an objective function is additively separable, the unconstrained optimization problem defined over it is also separable according to Equation (3.1).

Equation (3.2) has been commonly used to find subproblems in the literature (Omidvar et al., 2013, Mei et al., 2016, Hu et al., 2017, Omidvar et al., 2017, Sun et al., 2017, 2018, 2019). However, this definition does not represent all possible optimization separability types implied by Equation (3.1). Therefore, methods that are solely based on additive separability may not be able to find all subcomponents in a problem.

3.3 Decomposition methods

In this work, we used 3 well-studied and well-established decomposition methods, namely `DG` (Omidvar et al., 2013), `DG2` (Omidvar et al., 2017), and `RDG2` (Sun et al., 2017, 2018).

3.3.1 Differential Grouping (DG)

Proposed by Omidvar et al. (2013), the Differential Grouping (`DG`) method is one of the first and main decomposition strategies to examine and decompose continuous optimization problems automatically. Based on the partial additive separability definition presented in Equation (3.2), `DG` uses the following theorem to identify an interaction between two decision variables (Omidvar et al., 2013, Lopes et al., 2021):

Theorem 3.1. *Let $f(\boldsymbol{x})$ be an additively separable function. $\forall a, b_1 \neq b_2, \sigma \neq 0$, if the following condition holds*

$$\Delta_{\sigma,x_p}[f](\boldsymbol{x})|_{x_p=a,x_q=b_1} \neq \Delta_{\sigma,x_p}[f](\boldsymbol{x})|_{x_p=a,x_q=b_2} \tag{3.3}$$

implies

$$\Delta_{\sigma,x_p}[f](\boldsymbol{x})|_{x_p=a,x_q=b_1} - \Delta_{\sigma,x_p}[f](\boldsymbol{x})|_{x_p=a,x_q=b_2} \neq 0\,, \tag{3.4}$$

then x_p and x_q interact with each other, i.e. they are non-separable, where

$$\Delta_{\sigma,x_p}[f](\boldsymbol{x}) = f(\ldots, x_p + \sigma, \ldots) - f(\ldots, x_p, \ldots) \tag{3.5}$$

means the difference of $f(\boldsymbol{x})$ with respect to x_p with the interval σ.

In short, Theorem (3.1) determines that there is an interaction between decision variables x_p and x_q if Equation (3.4) yields a value different from 0. The proof of this theorem can be found in Omidvar et al. (2013).

The DG method considers the interaction of pairs of decision variables, x_p and x_q, for example. It analyzes if x_p has interaction with all other decision variables, x_q. If no decision variable interaction has been detected, it means x_p can be searched independently. Then, x_p is removed from the set of variables. Otherwise, if interactions are identified, all decision variables involved are grouped in the same set and an independent subproblem is defined over them. This process continues until there is no pair of decision variables to analyze (Omidvar et al., 2013, Lopes et al., 2021).

Theorem (3.1) is used to evaluate whether any pair of decision variables interact. To do that, DG initializes two vectors, **x** and **x**', with the lower bound of the decision variables. To verify the interaction between the decision variables x_p and x_q, the variable x_p from the vector **x**' is changed to its upper bound, then, the value of Δ_1 is computed as follows (Omidvar et al., 2013, Lopes et al., 2021):

$$\Delta_1 = f(\mathbf{x}) - f(\mathbf{x}')\,. \tag{3.6}$$

After that, q-th variables from **x** and **x**' are set to their middle value. The new points generated from these changes will be called **y** and **y**', respectively. Next, a new Δ is computed as follows (Omidvar et al., 2013, Lopes et al., 2021):

$$\Delta_2 = f(\mathbf{y}) - f(\mathbf{y}')\,. \tag{3.7}$$

Thus, an interaction between x_p and x_q is detected when the condition described in Equation (3.8) is true (Omidvar et al., 2013, Lopes et al., 2021).

$$|\Delta_1 - \Delta_2| > \epsilon\,, \tag{3.8}$$

where ϵ is a parameter that has to be defined by the user. The value of ϵ directly influences the interactions detected and its choice is not a trivial task. Another issue of this algorithm is that it may incorrectly identify subproblems since it skips through the decision variables for which

one interaction has already been identified, ignoring higher-order interactions (Omidvar et al., 2013, Lopes et al., 2021).

Even though DG is not the most used algorithm due to more function evaluations and its accuracy to identify the subproblems compared to the other decomposition strategies, its theoretical foundation is widely applied by other methods.

3.3.2 Differential Grouping 2 (DG2)

An improved version of the DG method was proposed in (Omidvar et al., 2017), called Differential Grouping 2 (DG2). DG2 is also based on Theorem (3.1). However, to deal with the disadvantages of its predecessor, DG2 does the following (Omidvar et al., 2013, Lopes et al., 2021):

1. Computes the raw interaction matrix which stores the values from $|\Delta_1 - \Delta_2|$ (see Equation (3.8)) for all pairs of decision variables;
2. Defines ϵ based on the raw interaction matrix by estimating the magnitude of round-off errors;
3. Defines an adjacency matrix based on the raw interaction matrix and the epsilon values;
4. Extracts the subproblems by analyzing the adjacency matrix.

The DG2 method has shown to be more efficient and presented greater grouping accuracy than DG. The authors in Lopes et al. (2021) point out that this version automatically defines a suitable threshold value ϵ. Furthermore, it can reuse sample points generated for detecting interactions.

3.3.3 Recursive Differential Grouping (RDG)

Recursive Differential Grouping (RDG) is a decomposition method proposed by Sun et al. (2017) that recursively analyzes the decision variable interaction. For that, it follows the Corollary next (Sun et al., 2017, Lopes et al., 2021):

Corollary 3.2. *Let $f(\boldsymbol{x})$, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ mutually exclusive subsets from the decision variables where $\boldsymbol{x}_1 \cap \boldsymbol{x}_2 = \emptyset$. If there two unit vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, two values l_1 and l_2 greater than 0, such that*

$$f(\boldsymbol{x} + l_1\boldsymbol{u}_1 + l_2\boldsymbol{u}_2) - f(\boldsymbol{x} + l_2\boldsymbol{u}_2) \neq f(\boldsymbol{x} + l_1\boldsymbol{u}_1) - f(\boldsymbol{x}) \tag{3.9}$$

there is some interaction of at least one pair of decision variables between sets $\boldsymbol{x}_1$ *and* $\boldsymbol{x}_2$*. Moreover,* $U_{\boldsymbol{x}_1}$ *is a subset of* $U_{\boldsymbol{x}}$ *such that any unit vector* $\boldsymbol{u}_1 = (u_1, \ldots, u_D) \in U_{\boldsymbol{x}}$*, where*

$$\boldsymbol{u}_i = 0, \ \textit{if}\ x_i \notin \boldsymbol{x}_1 \tag{3.10}$$

and the same idea is applied to $U_{\boldsymbol{x}_2}$*.*

According to Sun et al. (2017), the interaction between $\mathbf{x}_1$ and $\mathbf{x}_2$ subsets can be calculated by setting a point $\mathbf{x}$ with the lower bound of all decision variables. Then, $\mathbf{x}$' is defined by setting all decision variables with the lower bound and perturbing the variables from the subset $\mathbf{x}_1$ with the upper bound. After, the objective function value difference (δ_1) is calculated by

$$\delta_1 = f(\mathbf{x}) - f(\mathbf{x}') \, . \tag{3.11}$$

Perturbing the decision variables of the subset $\mathbf{x}_2$ from points $\mathbf{x}$ and $\mathbf{x}$' with the middle value between the lower and upper bounds, the points $\mathbf{y}$ and $\mathbf{y}$' will be generated. Thus, the objective function value difference is described by

$$\delta_2 = f(\mathbf{y}) - f(\mathbf{y}') \, . \tag{3.12}$$

If $|\delta_1 - \delta_2| > \epsilon$, then there is some interaction between the decision variables from subsets $\mathbf{x}_1$ and $\mathbf{x}_2$ where ϵ represents a control parameter from the RDG method to defined variable interaction. The ϵ parameter is defined according to the following Equation (Mei et al., 2016):

$$\epsilon = \alpha \times \min \{|f(\mathbf{x}_1)|, \ldots, |f(\mathbf{x}_n)|\} \, , \tag{3.13}$$

where n is the number of points randomly selected, and α is the controlling coefficient proposed by the authors in Mei et al. (2016). The parameters n and α must be defined before the decomposition process started.

Unlike the other decomposition methods, the RDG algorithm works recursively identifying the interaction between the decision variables. It starts from the first variable from the subset $\mathbf{x}_1$, and if there is no interaction, $\mathbf{x}_1$ is classified as a separable decision variable. If some interaction is detected, the remainder of the decision variables and the recursive identification of the interaction process continue.

RDG achieves a competitive grouping accuracy rate compared to the other decomposition methods (Sun et al., 2017). However, it requires a suitable parameter setting for the parameters n and α in order to compute the interaction threshold ϵ. The authors in Sun et al. (2018) propose a second version of the RDG. RDG2 computes values automatically for the ϵ parameter according

the formulation below:

$$\epsilon = \gamma \times \{|f(\mathbf{x})| + |f(\mathbf{x}')| + |f(\mathbf{y})| + |f(\mathbf{y}')|\} , \tag{3.14}$$

where γ is defined by the Equation next:

$$\gamma = (\upsilon \times \mu)/(1 - (\upsilon \times \mu)) , \tag{3.15}$$

where $\upsilon = \sqrt{D} + 2$, μ represents machine epsilon[2], and D is the dimensionality of the optimization problem.

3.4 Final discussion

In this work, we use `DG`, `DG2`, and `RDG2` as decomposition methods. `DG2` and `RDG2` are parameter-free. In `DG`, we defined $\epsilon = 0.1$ according to the related work (Omidvar et al., 2013). The implementation of the methods was extracted from the work presented by Lopes et al. (2021). The authors define a decomposition library called Continuous Optimization Problem Decomposition (COPD). Thus, we integrate our solver into the library COPD according to instructions presented in Lopes et al. (2021).

[2]It represents the difference between 1.0 and the next value representable by the floating-point. For C++, see epsilon function on numeric_limits.

Chapter 4

Diversity-driven migration strategy (DDMS)

Currently, many evolutionary algorithms have achieved excellent performance when promoting exploration and exploitation during evolution. Good examples are the *JADE* and *SHADE* algorithms and their variants. In Piotrowski and Napiorkowski (2018), the authors have shown *JADE/SHADE*-based methods outperform the vast majority of other metaheuristics (*DE*, *PSO*, *GA*, and variants) on some benchmark functions and real-world problems. Despite excellent results, these algorithms tend to lose diversity in the final stages of evolution (Arabas and Opara, 2019, Sudholt, 2020) or when the problem has many variables because of the curse of dimensionality (Mahdavi et al., 2015, LaTorre et al., 2015).

Therefore, we are interested in introducing a distributed structure for large-scale optimization problems where it is possible to ensure productive diversity within the islands through effective migration strategies. In this work, we propose new ideas to define the best moment to carry out the migration and what is the most appropriate composition of the individuals that will be migrated in order to increase the chances to restore diversity.

In this chapter, we will address two issues in our proposal: i) assess the loss of diversity in each dimension, and; ii) determine when to migrate. We have called this proposal of `DDMS` (Diversity-driven Migration Strategy).

4.1 Assessing the loss of diversity

In distributed algorithms, migration can be unnecessary or even out of time when done without criteria. A reasonable scenario would be to diversify when the population converges prematurely or stagnates.

4.1.1 Measuring premature convergence

In Yang et al. (2014), the authors propose a method to measure the premature convergence, as follows:

$$\mu_j^g = \frac{1}{\mathtt{NP}} \sum_{i=1}^{\mathtt{NP}} x_{i,j}^g \,, \tag{4.1}$$

$$\sigma_j^g = \sqrt{\frac{1}{\mathtt{NP}} \sum_{i=1}^{\mathtt{NP}} (x_{i,j}^g - \mu_j^g)^2} \,, \tag{4.2}$$

in which NP is the population size, μ_j^g and σ_j^g are the mean and the standard deviation of the population in the j-th dimension at generation g, respectively. The standard deviation (σ_j^g) measures the population diversity in the j-th dimension. When $\sigma_j^g \approx 0$, we have a strong indication that the j-th dimension has lost diversity. To denote whether the population has converged in the j-th dimension at the g-th generation, the flag $\bar{\tau}_j^g$ is defined as follows:

$$\bar{\tau}_j^g = \begin{cases} 1, & \text{if } \sigma_j^g \leq \omega_j^g \\ 0, & \text{otherwise} \end{cases} . \tag{4.3}$$

If σ_j^g is not greater than ω_j^g, $\bar{\tau}_j^g$ gets 1 to indicate the population has converged in the j-th dimension. The diversity is often large at the beginning of the evolutionary process and, consequently, the σ_j^g value is also large. Despite being problem-dependent, it usually takes a long time for the σ_j^g to drop to zero. Therefore, ω_j^g assumes a small value according to the condition below:

$$\omega_j^g = \min(T, \theta_j^g) \,. \tag{4.4}$$

The parameter T assumes a value close to zero (e.g. 10^{-3}). A variable θ_j^g is added to assess whether the j-th dimension had already converged on that same region in previous generations. The value of θ_j^g is defined as follows:

$$\theta_j^g = \begin{cases} |\mu_j^g - \mu_j^f| \times T, & \text{if } \sigma_j^g \leq T \\ T, & \text{otherwise} \end{cases} . \tag{4.5}$$

Suppose the population has converged in a given generation before g, which we can call generation g^-. Therefore, $\boldsymbol{\mu}^{g^-}$ vector is the population mean in generation g^-, regarding all subpopulations. If $\sigma_j^g \leq T$, the population has converged in the j-th dimension in the current generation g. We are interested in finding if this convergence in dimension j occurs in the same region as generation g^-. If $|\mu_j^g - \mu_j^{g^-}|$ is small, we assume that the population has converged to the same region. Observe in (4.5) that, in this case, θ_j^g is set to $|\mu_j^g - \mu_j^{g^-}| \times T$. Thus, as θ_j^g is less

than T, ω_j^g receives θ_j^g in (4.4). Note that while the algorithm remains converging to the same local minimum even after applying some diversification mechanism, ω_j^g value tends to decrease. Consequently, σ_j^g must be less than $\sigma_j^{g^-}$ to consider that the j-th dimension converged in (4.3). This schema allows the population to sufficiently exploit a revisited local optimum before using any diversification mechanism.

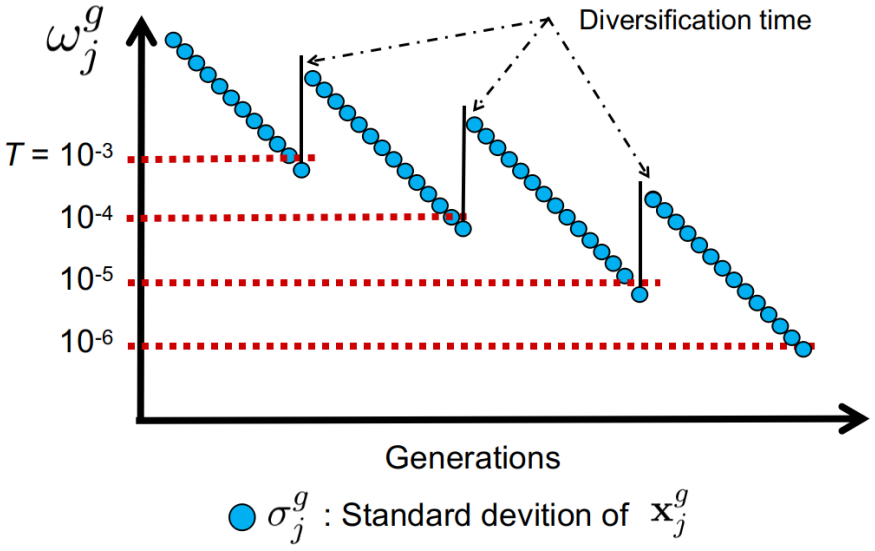


FIGURE 4.1: Example of convergence to the same region.

Figure 4.1 illustrates this schema considering a given dimension j. Note that, as the algorithm evolves, σ_j^g decreases until it surpass the initial threshold T (10^{-3}). In the first convergence, σ_j^g is equal to σ_j^f, then ω_j^g receives T and some diversification mechanism is activated. Assuming that the $\mathbf{x}_j^g$ remains converging to the same region, ω_j^g value continues to decrease, allowing the region to be better exploited before activating a new diversification. This strategy tends to be efficient when dealing with unimodal problems.

For multimodal problems, we need to explore efficiently new promising optimal solutions. For this purpose, a diversification mechanism can be activated always when the population converges to a different region than the previous one, which is when $|\mu_j^g - \mu_j^f|$ is large. In this case, ω_j^g is probably set to T. If σ_j^g is less than ω_j^g, a diversification is immediately triggered.

4.1.2 Measuring stagnation

Stagnation occurs when the population distribution does not change significantly for a while. This fact can be identified while the mean and the standard deviation remain unchanged for some successive generations (Yang et al., 2014). The authors in Yang et al. (2014) set the variable λ_j^g, which denotes the number of successive generations in which μ_j^g and σ_j^g remain

unchanged for the j-th dimension at the g-th generation. It is calculated as follows:

$$\lambda_j^g = \begin{cases} \lambda_j^{g-1} + 1, & \text{if } \mu_j^g = \mu_j^{g-1} \text{ and } \sigma_j^g = \sigma_j^{g-1} \\ 0, & \text{otherwise} \end{cases}, \tag{4.6}$$

in which $\lambda_j^g = 0$ in first generation. Another flag $\widehat{\tau}_j^g$ is defined to denote when the population has stagnated in the j-th dimension at the g-th generation, as follows:

$$\widehat{\tau}_j^g = \begin{cases} 1, & \text{if } \lambda_j^g \geq \texttt{UN} \\ 0, & \text{otherwise.} \end{cases}, \tag{4.7}$$

in which UN is an integer value. If $\lambda_j^g \geq \texttt{UN}$ then $\widehat{\tau}_j^g = 1$, which indicates that the population can be stagnated in the j-th dimension. In Yang et al. (2014), the authors show the larger the population size, the more generations it will take for the population to enter a stable stagnation state. Therefore, they use UN = NP, in which NP is the population size.

The authors in Yang et al. (2014) apply these approaches to identify when a diversification mechanism should be utilized. In this work, we define three diversification mechanisms based on migration schemes between islands in distributed computing environments. Thus, we use these approaches to identify when to migrate.

4.2 Determining migratory frequency

Consider a population $P = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\texttt{NP}}\}$ in which NP is the population size. Each individual is represented by a vector $\mathbf{x}_i = \{x_{i,j}, \ldots, x_{i,\texttt{D}}\}, i = 1, 2, \ldots, \texttt{NP}$ and $j = 1, 2, \ldots, \texttt{D}$, in which D is the number of dimensions. Figure 4.2 illustrates the process of identifying the population's convergence and stagnation as presented above.

This example shows convergence did not happen in dimensions 1 to 4 since the standard deviation (σ) is still high. That is, these dimensions still preserve a certain diversity. However, note that σ_5^g (0.0007) is less than T (10^{-3}). Since the difference between the mean (μ_5^g) and the value of the mean of the last convergence (μ_5^f) is not small, ω_5^g assumes the value of T. This means that the region of the last convergence is significantly different from the current one. As the σ_5 is less than ω_5, τ_5 receives 1, indicating that this dimension needs to be diversified.

Regarding stagnation, we can observe that, in dimensions 2 and 4, the values of μ and σ have not changed for 5 generations ($\lambda = 5$). Since UN = NP, $\widehat{\tau}$ assumes 1, which means that we can consider that both dimensions may be trapped. The authors in Yang et al. (2014) define the

	1	2	3	4	5
x_1	2.511	4.222	3.163	2.130	0.241
x_2	2.701	4.012	3.510	5.427	0.242
x_3	1.321	4.312	3.273	1.142	0.241
x_4	4.532	4.221	3.413	1.741	0.240
x_5	4.531	4.212	3.493	1.784	0.240
μ_j^g	3.119	4.196	3.370	2.445	0.241
σ_j^g	1.246	0.099	0.133	1.525	**0.0007**
Convergence					
μ_j^f	2.151	4.196	3.301	2.445	1.821
θ_j^g	10^{-3}	10^{-3}	10^{-3}	10^{-3}	$1.6x10^{-3}$
ω_j^g	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
$\bar{\tau}_j^g$	0	0	0	0	**1**
Stagnation					
λ_j^g	2	5	0	5	**0**
$\widehat{\tau}_j^g$	0	**1**	0	**1**	0
Enhancement					
τ_j^g	0	**1**	0	**1**	**1**

FIGURE 4.2: Example of identifying the population's convergence and stagnation, considering $\texttt{NP} = 5$ and $\texttt{D} = 5$.

follow condition:

$$\tau_j^g = \begin{cases} 1, & \text{if } \bar{\tau}_j^g = 1 \text{ or } \widehat{\tau}_{j,g} = 1 \\ 0, & \text{otherwise} \end{cases}. \tag{4.8}$$

That is, if $\bar{\tau}_j$ or $\widehat{\tau}_j$ is equal to 1, the j-th dimension needs some diversification mechanism must be used. In this work, we utilize migrations between islands to improve diversity.

After identifying whether the population has converged or stagnated in the j-th dimension, some rules are checked to decide if migration should be triggered.

- **Rule 1:** The authors in Yang et al. (2014) show immediate diversifying of the converged or stagnated dimension identified by the Equation (4.8) may deteriorate the search, thus slowing up convergence. Therefore, migration occurs when all dimensions have lost diversity, as shown below:

$$\Gamma 1^g = \begin{cases} 1, & \text{if } \texttt{NDIV = D} \\ 0, & \text{otherwise} \end{cases}, \tag{4.9}$$

in which NDIV is the number of dimensions that need some diversity, being calculated as $\texttt{NDIV} = \sum_{j=1}^{\texttt{D}} \tau_j^g$.

- **Rule 2:** For high-dimensional problems, a large number of function evaluations (generations) would be required for the algorithm to trigger **Rule 1** ($\Gamma 1^g$). To alleviate this issue, migration occurs with a small probability in any dimension j according to the following condition (Yang et al., 2014):

$$\Gamma 2^g = \begin{cases} 1, & \text{if } \tau_j^g = 1 \quad \text{and} \quad rand() < c \\ 0, & \text{otherwise} \end{cases}, \tag{4.10}$$

in which c is a small constant number, e.g., $c = 10^{-3}$ (Yang et al., 2014). The $rand()$ function generates a uniformly distributed random number within $[0, 1]$.

- **Rule 3:** In a distributed algorithm, inopportune migrations happen because most proposals use fixed or probabilistic migrations. For this reason, we propose a third rule, as shown below:

$$\Gamma 3^g = \begin{cases} 1, & \text{if } \Phi \geq \Psi \\ 0, & \text{otherwise} \end{cases}, \tag{4.11}$$

in which

$$\Phi = \frac{\texttt{NDIV}}{\texttt{D}} \quad \text{and} \quad \Psi = 1 - \left(\frac{\texttt{NFE}}{\texttt{NFE}^{max}} \right). \tag{4.12}$$

In this definition, D is the number of decision variables, and NDIV is the number of dimensions that need some diversity. Then, Φ represents the proportion of dimensions that must be diversified. NFE is the current number of function evaluations, while $\texttt{NFE}^{max}$ is the maximum number of function evaluations. We define a metric that helps to estimate when diversity is least likely to be lost. Diversity loss is much less likely to occur early in the evolutionary process, so the value of Ψ is higher at this stage. Therefore, migrations only happen early in the evolutionary process if the value of Φ is massive.

As the algorithm evolves, convergence and stagnation become more likely to happen, so Ψ will become smaller. Thus, more migrations can occur at the end of the evolutionary process even if the Φ value is small.

Figure 4.3 shows that this approach can identify situations of diversification that the authors in Yang et al. (2014) do not cover. The green line represents the Ψ values as the algorithm evolves. The points are the proportion of dimensions that must be diversified

(Φ) in that generation[1]. The red points are generations in which a migration must not occur and the blue points are generations in which a migration must occur, according to the **Rule 3** ($\Gamma 3^g$).

According to this proposal, migration can occur in the first half of evolution since the proportion of variables that must be diversified (Φ) is not too small. On the other hand, the probability for dimensions to converge or stagnate increases after half the evolution. Thus, even with the small value of Φ, more migrations can occur to improve diversity at the end of the evolutionary process.

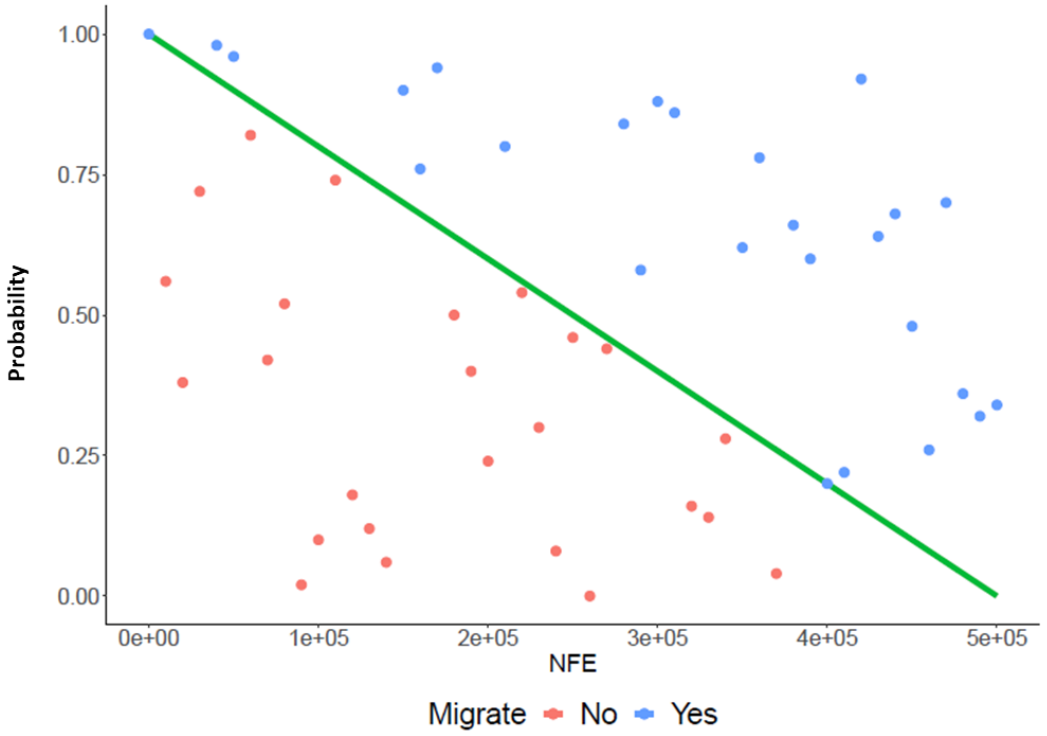


FIGURE 4.3: **Rule 3:** When to migrate. The green line represents the Ψ values as the algorithm evolves. Red points are generations in which a migration must not occur and blue points are generations in which a migration must occur.

The rules for determining when to migrate are summarized as follows:

$$\Gamma^g = \begin{cases} 1, & \text{if } \Gamma 1^g = 1 \text{ or } \Gamma 2^g = 1 \text{ or } \Gamma 3^g = 1 \\ 0, & \text{otherwise} \end{cases} . \tag{4.13}$$

After each generation g, this procedure generates the Γ^g value. If $\Gamma^g = 1$, then the migration must happen; otherwise, the migration is discarded.

[1] For simplicity, we generate these points randomly.

4.3 Evolving subpopulation

Any optimization algorithm can be used to evolve subpopulations within islands, such as *DE*, *GA*, *PSO*, among others. In this work, we use the traditional version of the *DE* (Storn and Price, 1997) and an improved version of the *DE* (*SHADE* (Tanabe and Fukunaga, 2014)). This section briefly describes these algorithms.

4.3.1 *DE*

Proposed by Storn and Price (1997), *DE* is a well-known population-based metaheuristic. In each generation g, *DE* generates a mutant vector $\mathbf{v}_i^g$ by employing a mutation operation on each candidate solution $\mathbf{x}_i^g$. Some of the most frequently used mutation strategies in *DE* are listed as follows:

1. *DE*/rand/1
$$\mathbf{v}_i^g = \mathbf{x}_{r_1}^g + F \times (\mathbf{x}_{r_2}^g - \mathbf{x}_{r_3}^g)\,, \tag{4.14}$$
2. *DE*/best/1
$$\mathbf{v}_i^g = \mathbf{x}_{best}^g + F \times (\mathbf{x}_{r_1}^g - \mathbf{x}_{r_2}^g)\,, \tag{4.15}$$
3. *DE*/current-to-best/1
$$\mathbf{v}_i^g = \mathbf{x}_i^g + F \times (\mathbf{x}_{best}^g - \mathbf{x}_i^g) + F \times (\mathbf{x}_{r_1}^g - \mathbf{x}_{r_2}^g)\,, \tag{4.16}$$
4. *DE*/rand-to-best/1
$$\mathbf{v}_i^g = \mathbf{x}_{r_i}^g + F \times (\mathbf{x}_{best}^g - \mathbf{x}_{r_i}^g) + F \times (\mathbf{x}_{r_2}^g - \mathbf{x}_{r_3}^g)\,. \tag{4.17}$$

The indices r_1, r_2 and r_3 are mutually exclusive integers randomly generated within the range $\{1,2,\ldots,\texttt{NP}\}$. x_{best}^g is the individual with the best fitness value at the generation g. The scaling factor $F \in [0,1]$ controls the magnitude of the differential mutation operator.

After that, the mutant vector $\mathbf{v}_i^g$ is crossed with the target vector $\mathbf{x}_i^g$ and a trial vector $\mathbf{u}_i^g$ is generated. Equation (4.18) describes a commonly used binomial crossover operator in *DE*.

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } rand() \leq CR \text{ or } j = j_{rand} \\ x_{i,j}^g, & \text{otherwise} \end{cases}\,. \tag{4.18}$$

The $rand()$ function generates a uniformly distributed random number within $[0, 1]$ and j_{rand} is a decision variable index that is randomly selected from $[1, \mathtt{D}]$. Finally, $CR \in [0, 1]$ is the crossover rate.

Then, a selection process determines which vectors must survive for the next generation. Equation (4.19) describes a common selection operator which compares each individual $\mathbf{x}_i^g$ against its corresponding trial vector $\mathbf{u}_i^g$, keeping the better vector in the population.

$$\mathbf{x}_i^{g+1} = \begin{cases} \mathbf{u}_i^g, & \text{if } f(\mathbf{u}_i^g) \leq f(\mathbf{x}_i^g) \\ \mathbf{x}_i^g, & \text{otherwise} \end{cases} . \tag{4.19}$$

4.3.2 *JADE*

In Zhang and Sanderson (2009b), the authors propose an improved version of the *DE*, called *JADE* (adaptive differential evolution with optional external archive). They introduce three new features: I) a new mutation strategy (*DE*/current-to-*p*best/1), II) an external archive, and III) an adaptive control of the F and CR values. These features are briefly described below.

I. **New mutation strategy**: the mutation strategy *current-to-best* shown in Equation (4.16) drives the generation of mutant vectors towards the best individual of the population. A side effect of this strategy is that the search can converge quickly to a local optimum. Although this behavior is useful for unimodal problems, this premature convergence can mean a poor performance on multimodal problems (Zhang and Sanderson, 2009b, Tanabe and Fukunaga, 2013). Because of that, the *JADE* algorithm presents a new mutation strategy called current-to-*p*best/1, in which a new parameter p is added to adjust the degree of greediness of the mutation strategy. Equation (4.20) summarizes this feature.

$$\mathbf{v}_i^g = \mathbf{x}_i^g + F_i \times (\mathbf{x}_{pbest}^g - \mathbf{x}_i^g) + F_i \times (\mathbf{x}_{r_1}^g - \mathbf{x}_{r_2}^g) , \tag{4.20}$$

in which $\mathbf{x}_{pbest}^g$ is an individual randomly selected from the $\mathtt{NP} \times p$ best individuals in the g-th generation, with $p \in [0, 1]$. The p parameter controls the degree of greediness and, hence, the balance between exploitation and exploration (small p means more greedy).

II. **External archive**: *JADE* also uses an archive feature to maintain diversity. Parent vectors $\mathbf{x}_i^g$ which are discarded in Equation (4.19) are stored in an archive $\mathbf{A}$. In Equation (4.20), the individual $\mathbf{x}_{r_2}^g$ is selected from union of the population $\mathbf{P}$ and the archive $\mathbf{A}$ ($\mathbf{P} \cup \mathbf{A}$). The maximum size of the archive, $\mathtt{NPA}$, is the same as the population ($\mathtt{NPA} = \mathtt{NP}$). If the size of the archive exceeds $\mathtt{NPA}$, individuals are randomly deleted to make space for the newly inserted individuals.

III. **Adaptive control of F and CR values**: In *JADE*, each individual $\mathbf{x}_i$ has its own CR_i and F_i parameters in Equations (4.18) and (4.20). At the beginning of each generation g, the crossover probability CR_i of each individual $\mathbf{x}_i$ is generated according to a normal distribution ($\mathcal{N}$) with mean μ_{CR} and variance $\sigma^2 = 0.1$. Similarly, at each generation g, the mutation factor F_i of each individual $\mathbf{x}_i$ is independently generated according to a Cauchy distribution ($\mathcal{C}$) with mean μ_F and variance $\sigma^2 = 0.1$. Equations (4.21) and (4.22) demonstrate such operations.

$$CR_i = \mathcal{N}(\mu_{CR}, 0.1)\ , \tag{4.21}$$

$$F_i = \mathcal{C}(\mu_F, 0.1)\ . \tag{4.22}$$

If CR_i is out the range of $[0, 1]$, then it is rounded to the nearest bound (0 or 1). If $F_i > 1$, then F_i receives 1. If $F_i \leq 0$, then Equation (4.22) is repeatedly executed until the generation of a valid value. In each generation, CR_i and F_i values that succeed in generating a trial vector $\mathbf{u}_i^g$ which is better than the parent vector $\mathbf{x}_i^g$ are stored in the $\mathbf{SCR}$ and $\mathbf{SF}$ vectors, and at the end of the generation, μ_{CR} and μ_F are updated as:

$$\mu_{CR} = (1 - c) \times \mu_{CR} + c \times \mu_A(\mathbf{SCR})\ , \tag{4.23}$$

$$\mu_F = (1 - c) \times \mu_F + c \times \mu_L(\mathbf{SF})\ . \tag{4.24}$$

In first generation, μ_{CR} and μ_F are both initialized to 0.5. Zhang and Sanderson (2009b) define the learning rate c equal to 0.1. Finally, $\mu_A(\cdot)$ is an arithmetic mean, and $\mu_L(\cdot)$ is a Lehmer mean which is computed as:

$$\mu_L = \frac{\sum_{k=1}^{|\mathbf{SF}|}(SF_k)^2}{\sum_{k=1}^{|\mathbf{SF}|} SF_k}\ . \tag{4.25}$$

Zhang and Sanderson (2009b) explain such proposals. The idea is to propagate to the next generations the values of μ_{CR} that tend to generate individuals more likely to survive and, thus, guide the generation of new values of CR_i's. The adaptation of μ_F uses the Cauchy distribution because it is more useful to diversify the mutation factors and thus avoid premature convergence. In addition, the adaptation of μ_F using the Lehmer mean gives preference to propagate larger successful mutation factors, which tends to improve the progress rate.

4.3.3 *SHADE*

Tanabe and Fukunaga (2013) propose an improved version of *JADE* which uses a different

parameter adaptation mechanism. In Success-History based Adaptive *DE* (*SHADE*), the mean values of $\mathbf{SCR}$ and $\mathbf{SF}$ for each generation are stored in $\mathbf{MCR}$ and $\mathbf{MF}$ vectors, respectively. Thus, *SHADE* maintains a diverse set of parameters to drive control parameter adaptation. Now, Equations (4.21) and (4.22) become:

$$CR_i = \mathcal{N}(MCR_r, 0.1)\,, \tag{4.26}$$

$$F_i = \mathcal{C}(MF_r, 0.1)\,. \tag{4.27}$$

In contrast to *JADE*, which uses a single pair (μ_{CR}, μ_F) to guide parameter adaptation, the control parameters CR_i and F_i used by each individual $\mathbf{x}_i$ are generated by selecting an index r randomly from $\{1, \mathtt{H}\}$, in which $\mathtt{H}$ is the maximum number of entries that can be saved in $\mathbf{MCR}$ and $\mathbf{MF}$. If CR_i or F_i exceeds the range $[0, 1]$, the same procedures described for *JADE* are utilized. The contents of memories $\mathbf{MCR}$ and $\mathbf{MF}$ are updated as follows:

$$MCR_k^{g+1} = \begin{cases} \mu_{WA}(\mathbf{SCR}), & \text{if } |\mathbf{SCR}| \neq 0 \\ MCR_k^g, & \text{otherwise} \end{cases}, \tag{4.28}$$

$$MF_k^{g+1} = \begin{cases} \mu_{WL}(\mathbf{SF}), & \text{if } |\mathbf{SF}| \neq 0 \\ MF_k^g, & \text{otherwise} \end{cases}, \tag{4.29}$$

in which $k \in \{1, 2, \ldots, \mathtt{H}\}$ is the position in the memory to update. At the beginning of the search, k is initialized to 1 and both $\mathbf{MCR}$ and $\mathbf{MF}$ vectors are initialized to 0.5. When a new element is inserted into the history, k is incremented. If $k > \mathtt{H}$, k is reset to 1. In generation g, the k-th element in the memory is updated. When all individuals in generation g are unable to generate at least one trial vector $\mathbf{u}_i$ which is better than the parent $\mathbf{x}_i$, i.e., $|\mathbf{SCR}| = |\mathbf{SF}| = 0$, the memory is not updated. To prevent MCR_k converges to a small value, the authors replace the arithmetic mean with the following weighted mean:

$$\mu_{WA}(\mathbf{SCR}) = \sum_{k=1}^{|\mathbf{SCR}|} w_k \times SCR_k\,, \tag{4.30}$$

in which

$$w_k = \frac{\Delta f_k}{\sum_{k=1}^{|\mathbf{SCR}|} \Delta f_k}\,, \tag{4.31}$$

in which $\Delta f_k = |f(\mathbf{u}_k^g) - f(\mathbf{x}_k^g)|$. Thus, the amount of improvement is used in order to influence the parameter adaptation. w_k is also used to compute the weighted Lehmer mean $\mu_{WL}(\mathbf{SF})$, as follows:

$$\mu_{WL}(\mathbf{SF}) = \frac{\sum_{k=1}^{|\mathbf{SF}|} w_k \times (SF_k)^2}{\sum_{k=1}^{|\mathbf{SF}|} w_k \times (SF_k)}\,. \tag{4.32}$$

Also, *SHADE* implements a schema to adapt the parameter p in the current-to-$p$best/1 mutation strategy presented in Equation (4.20). In each generation g, an individual $\mathbf{x}_i$ has an associated p_i, which is set according to the equation below:

$$p_i \quad = rand[p_{min}, p_{max}] \,, \tag{4.33}$$

in which $p_{min} = {}^{2}/_{\text{NP}}$ so that, when the $p$best individual is selected, at least 2 individuals are part of the set. Related to the maximum value, the authors use $p_{max} = 0.2$ as proposed by Zhang and Sanderson (2009a).

4.4 Final discussion

This chapter establishes the rules to determine when to migrate considering population convergence and stagnation. In the next chapter, we present an efficient strategy to compute convergence and stagnation based on a framework to detect anomaly using the cumulative proximity between data samples. This strategy updates recursively the statistics (mean and standard deviation) in each dimension as if it were a data stream. Thus, the proposal made by Yang et al. (2014) can be implemented by incrementally modeling a non-parametric data distribution for each decision variable.

Also, we introduce a new migratory policy based on a centralization-clustering approach in order to keep a recent migration history. This history is used to generate clusters with similar and good fitness individuals, but also identify the changes in the parameters of the data distribution.

Chapter 5

TEDA-based approach to defining the migratory policy

5.1 Defining migratory policy

From now on, we are interested in defining how migration should be done. Figure 5.1 illustrates the proposed strategy. The idea is to migrate an individual not too similar to the island population that has lost diversity. Meanwhile, we seek to generate multiple migration options considering the recent migration history.

Consider that Islands 1 and 2 identify the need for migration according to a criterion (for example, the rules presented in the last chapter). In this case, they send a vector with the best individual to a pool (step 1). In this pool, such a best individual will be clustered into a cloud together with similar individuals. In Bezerra et al. (2016), the authors introduce a new evolving clustering algorithm called *TEDA*-Class. Using a *TEDA*-based (Typicality and Eccentricity Data Analysis) approach, they implement a clustering algorithm that evolves in an iterative and autonomous way, without the need for constant adjustments in its parameters, which is very useful to deal with situations in which changes in data distributions are not usually known *a priori*. In this work, we adjusted the *TEDA*-Class to group similar individuals arriving at the pool.

After grouping, the algorithm chooses an individual to be migrated (**MIG**). Preferably, this individual should be different from the population distribution of the island (step 2). Subsequently, **MIG** is then sent to the island (step 3). Finally, the island selects the values in the variables that should be diversified considering the $\boldsymbol{\tau}^g$ vector (step 4). Next, the empty dimensions of $\boldsymbol{\beta}^g$ are filled regarding the best individual of the island, then generating a new individual that replaces a random individual into the subpopulation.

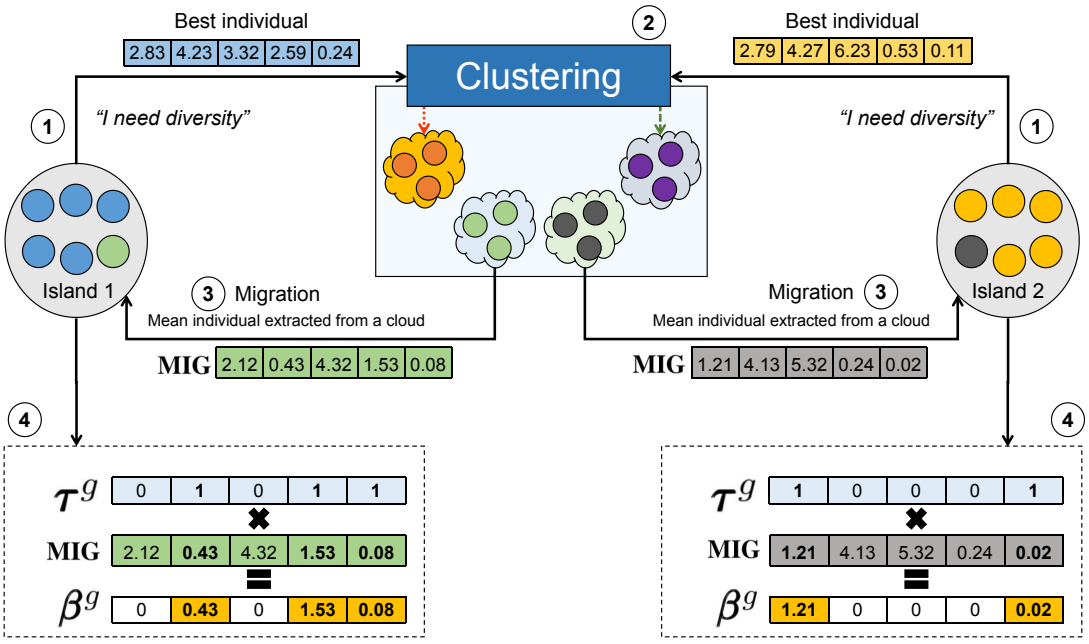


FIGURE 5.1: Illustration of a distributed EA with the proposed migratory policy. Circles into the island represent the individuals of the population. The values in the small circles and the black arrows represent the steps/sequence of the events. The dashed rectangles show actions taken on the islands. The rectangle with a straight line houses the clouds obtained by the clustering algorithm.

In contrast to traditional migration models that involve direct communication policies, this proposal introduces an island model based on a clustering framework. This practice can provide straight control of exploration and exploitation to maintain diversity. To this end, we seek to increase the probability that the migrated individual will be valuable in promoting diversity. Meanwhile, the migrated individual must have a quality track record, which helps the island escape stagnation and move into promising new regions.

We can combine this proposal with the DDMS presented in the previous chapter, as shown in Algorithm 1. The algorithm has as input parameters: the maximum number of function evaluations (NFE^{max}); the size of population (NP); the number of dimensions (D); and the $role$, which can be "pool" or "island". The subpopulation is initialized randomly, and an objective function $f(\cdot)$ evaluates solutions (line 1). At any given iteration (line 3), the algorithm executes until met the stopping condition. Generally, the algorithm stops when NFE^{max} is reached or the optimal solution is found. In each generation g, the number of function evaluations (NFE) is computed according to the number of individuals in the population (line 4).

A typical island runs an optimizer that guides the evolutionary process (line 6). In our experiments, we choose *DE* (Storn and Price, 1997) and *SHADE* (Tanabe and Fukunaga, 2014) as optimizers. After that, the island identifies in which dimensions the population has lost diversity, as shown in subsection 4.1 (lines 7). Finally, the rules described in subsection 4.2 are

applied to define if a migration must occur or not (line 8). If a migration must occur (line 10), the island sends its best individual to the pool (line 11).

In a loop, the pool is always available to receive the best individual (**best_ind**) of any island that needs diversity (line 17). In this case, the clustering algorithm presented in subsection 5.3 is used to group such individuals in *data clouds* (line 18). After that, the average individual (**MIG**) is extracted from any *data cloud* in which the **best_ind** is considered an outlier (line 19). Subsequently, this selected individual (**MIG**) is then sent to the island (line 20). Once the migrated individual (**MIG**) has been received (line 12), the island selects the values in the dimensions that should be diversified, according to the anomaly detection of *TEDA* described in the subsection 5.2. The empty dimensions of $\boldsymbol{\beta}^g$ are filled by the best individual, and this new individual replaces a random solution, different from the best one (lines 13–14). Lastly, g is incremented, and a new generation begins (line 21).

Algorithm 1 General structure of the DDMS-TEDA.

NFE^{max}: Maximum number of function evaluations; NP: size of population; D: Number of dimensions; $role$: define the role of the island;

```
1:  Initialize subpopulation with NP individuals
2:  g ← 1                                                          ▷ First generation
3:  while the stopping condition is not met do
4:      NFE ← g × NP                                   ▷ Number of function evaluations
5:      if role == "island" then                                              ▷ Island
6:          Execute optimizer
7:          Identify premature convergence and stagnation using equations (4.3) and (4.7)
8:          Determine if a migration must occur using equation (4.13)
9:
10:         if migration must occur then                              ▷ "Need diversity"
11:             Send the best individual, best_ind, to the pool
12:             Receive the migrated individual MIG
13:             Store in β_g the dimensions that should be diversified
14:             Replace a random individual by composition of β_g and best_ind
15:     else if role == "pool" then                                             ▷ Pool
16:         while migration request has arrived do
17:             Receive the best individual, best_ind, from island
18:             Perform clustering of best_ind
19:             Extract the average individual, MIG, of one of the clouds in which best_ind is an outlier
20:             Send MIG to the island
21:     g ← g + 1                                                 ▷ Update generation
```

5.2 *TEDA*

TEDA is a framework proposed by Angelov (2014) which is specially used to detect anomaly. Basically, *TEDA* incrementally models a non-parametric data distribution based only on the cumulative proximity between data samples. Consider a D-dimensional input vector $\mathbf{x}^k$ in the timestamp k. The cumulative proximity $\pi(\cdot)$ of $\mathbf{x}^k$ in relation to all existing data samples is

calculated as:

$$\pi^k(\mathbf{x}) = \sum_{i=1}^{k} \text{dist}(\mathbf{x}^k, \mathbf{x}^i) \,, \tag{5.1}$$

in which $\text{dist}(\mathbf{x}^k, \mathbf{x}^i)$ is the distance between data points $\mathbf{x}^k$ and $\mathbf{x}^i$; k is the timestamp when the data point $\mathbf{x}$ is sampled. For anomaly detection, *TEDA* works with two concepts: eccentricity and typicality. Eccentricity is a measure of the dissimilarity between the data point $\mathbf{x}^k$ in relation to all the data points received until timestamp k. On contrary, the typicality represents how typical an arbitrary data point $\mathbf{x}^k$ is in relation to all the data points received until the timestamp k. For the case of Euclidean distance, eccentricity can be calculated recursively as follows (Angelov, 2014):

$$\xi^k(\mathbf{x}) = \frac{1}{k} + \frac{(\boldsymbol{\mu}^k - \mathbf{x}^k)^T(\boldsymbol{\mu}^k - \mathbf{x}^k)}{k \times (\sigma^2)^k} \,, \tag{5.2}$$

in which $\boldsymbol{\mu}^k$ and $(\sigma^2)^k$ are the average and variance, respectively, that can also be recursively updated as follows (Angelov, 2014):

$$\boldsymbol{\mu}^k = \frac{k-1}{k} \times \boldsymbol{\mu}^{k-1} + \frac{\mathbf{x}^k}{k} \,, \tag{5.3}$$

$$(\sigma^2)^k = \frac{k-1}{k} \times (\sigma^2)^{k-1} + \frac{1}{k-1} \times ||\boldsymbol{\mu}^k - \mathbf{x}^k||^2 \,, \tag{5.4}$$

with $k \geq 1$. When $k = 1$, $\boldsymbol{\mu}^k = \mathbf{x}^k$ and $(\sigma^2)^k = 0$. The normalized eccentricity $\zeta^k(\mathbf{x})$ and typicality $T^k(\mathbf{x})$ can be obtained as follows:

$$\zeta^k(\mathbf{x}) = \frac{\xi^k(\mathbf{x})}{2} \,, \tag{5.5}$$

$$T^k(\mathbf{x}) = \frac{1 - \xi^k(\mathbf{x})}{k-2} \,, \tag{5.6}$$

with $k \geq 2$. The authors in Bezerra et al. (2016) illustrate typicality and eccentricity concepts with the Figure 5.2. Observe that the data point **A** is more distant from the data set than the data point **B**. Therefore, **A** has higher eccentricity and lower typicality than **B** (Bezerra et al., 2016).

The normalized eccentricity $\zeta^k(\mathbf{x})$ can be used to determine if $\mathbf{x}^k$ is an outlier in relation to the data set. For this purpose, $\zeta^k(\mathbf{x})$ becomes a threshold based on the well known *Chebyshev* inequality (Saw et al., 1984)

$$\zeta^k(\mathbf{x}) > \frac{m^2 + 1}{2 \times k} \,, \tag{5.7}$$

in which m is a constant value that is greater than 0. This inequality can be explained in the following way: if the actual sample $\mathbf{x}^k$ is $m \times \sigma$ away from the average, then it is considered an outlier in relation to the data set. Therefore, m represents how many standard deviations

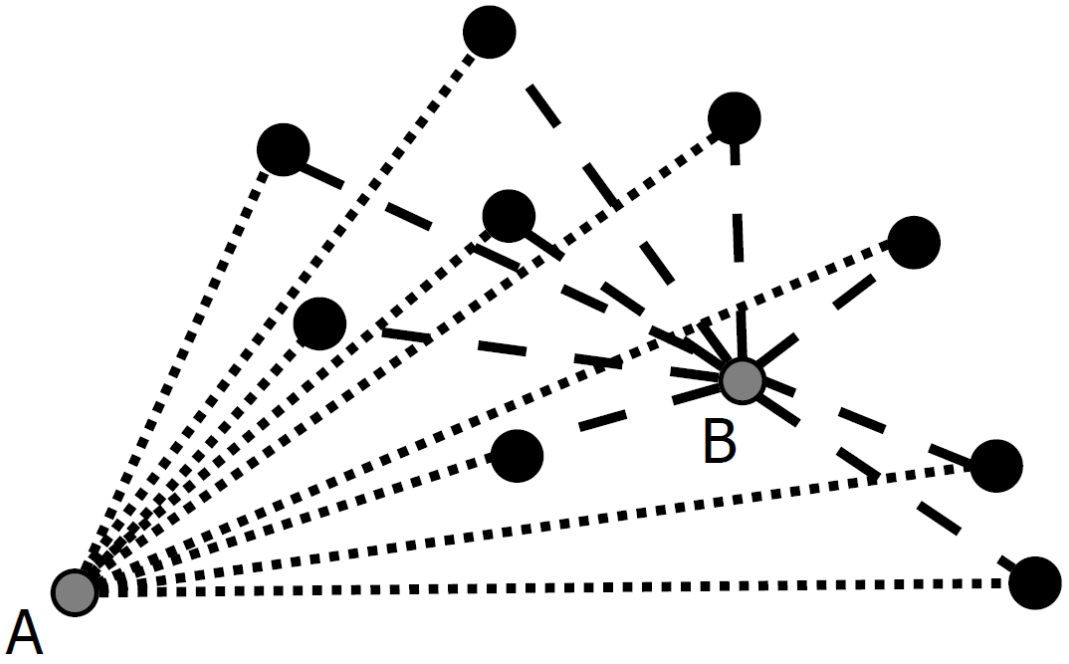


FIGURE 5.2: Illustration of Typicality and Eccentricity concepts in *TEDA* (Bezerra et al., 2016)

distant from the average a data sample should be in order to be considered an outlier (Angelov, 2014). We use the robustness to anomaly detection of *TEDA* to extract the individual that will be migrated.

5.3 Adapted *TEDA*-Cloud

In this section, we detail the implementation of the functions to perform the clustering of the best individual, **`best_ind`**, received from the island and extract the average individual, **`MIG`**, of one of the clouds in which **`best_ind`** is an outlier (Algorithm 1, lines 18–19).

We create and update clusters based on the *TEDA*-Cloud algorithm proposed in Bezerra et al. (2016). The approach generates granular data structures called *data clouds*, whose main characteristic is that they do not have predefined shapes or boundaries as traditional data clusters. In *TEDA*-Cloud, *data clouds* are being formed as the data arrives by considering common properties among the data samples. This way, the set of *data clouds* directly describes all previous data samples. Another characteristic of *TEDA*-Cloud is the possibility to represent the data sets in terms of fuzzy membership. An example is that a particular data sample can belong to all *data clouds* with different membership degrees, with values between $[0, 1]$.

The algorithm derives each equation from *TEDA* to a generalized form, in which each *data cloud* is an independent data set. Then, it determines the membership of each read data sample to each existing *data cloud*, based on Equation (5.7). Let $\mathcal{DC}_i$, $i = 1, \ldots, \mathtt{NC}$, be the set of *data*

clouds, in which NC is the number of clusters. A prototype of $\mathcal{DC}_i$ is defined by the following parameters, which are updated every time a new data point arrives at the pool:

- s_i^k: number of data samples;
- $\boldsymbol{\mu}_i^k$: the average, which represents the center of *data cloud*;
- $(\sigma^2)_i^k$: variance;
- $\xi_i(\mathbf{x}^k)$, $\zeta_i(\mathbf{x}^k)$: eccentricity and normalized eccentricity;
- $T_i(\mathbf{x}^k)$: normalized typicality;

The first cluster $\mathcal{DC}_1$ is created when the first data sample $\mathbf{x}_1$ arrives. In this case, the parameters are initialized as follows:

$$\text{NC} = 1, \quad s_1^1 = 1, \quad \boldsymbol{\mu}_1^1 = \mathbf{x}_1, \quad (\sigma^2)_1^1 = 0\,. \tag{5.8}$$

It is worth mentioning that typicality and eccentricity can only be calculated with at least two data samples. Figure 5.3 illustrates three *data clouds* ($\mathcal{DC}_1$, $\mathcal{DC}_2$ and $\mathcal{DC}_3$) after k observations. For the sake of simplicity, structures are represented as circles. However, *data clouds* do not have specific shapes because *TEDA*-Cloud identifies the true distribution of the data.

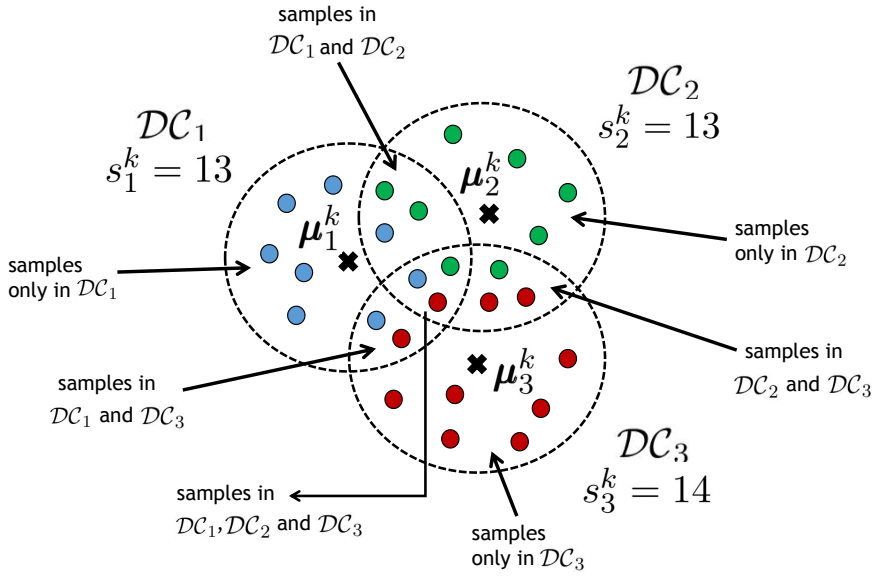


FIGURE 5.3: *Data clouds* $\mathcal{DC}_1$, $\mathcal{DC}_2$ and $\mathcal{DC}_3$ after k observations.

The number of data points that belong to *data clouds* $\mathcal{DC}_1$, $\mathcal{DC}_2$ and $\mathcal{DC}_3$ is $s_1^k = 13$, $s_2^k = 13$ and $s_3^k = 14$, respectively. Note that there are points that are in multiple clusters. As a fuzzy-based approach, *TEDA*-Cloud allows a given data sample can simultaneously belong to

more than one *data cloud*, which generates an intersection region between two or more clusters. Besides, the averages $\boldsymbol{\mu}_1^k$, $\boldsymbol{\mu}_2^k$ and $\boldsymbol{\mu}_3^k$ graphically represent the center of the *data clouds* $\mathcal{DC}_1$, $\mathcal{DC}_2$ and $\mathcal{DC}_3$, respectively. Finally, the variances $(\sigma^2)_1^k$, $(\sigma^2)_2^k$ and $(\sigma^2)_3^k$ represent the data spread in *data clouds* $\mathcal{DC}_1$, $\mathcal{DC}_2$ and $\mathcal{DC}_3$, respectively (Bezerra et al., 2016).

We adapt *TEDA*-Cloud to work into a pool that receives the best individual from a given island whenever a migration is triggered, according to the rules previously presented. Basically, for each individual $\mathbf{x}^k$ which arrives at the pool, *TEDA*-Cloud computes its eccentricity $\zeta_i(\mathbf{x}^k)$ in relation to the *data cloud* $\mathcal{DC}_i$. If $\zeta_i(\mathbf{x}^k)$ is high, we assume that $\mathbf{x}^k$ is considerably different from the data samples belonging to $\mathcal{DC}_i$, therefore, it is not necessary to carry out any updates to the cloud structure. Otherwise, we assume that $\mathbf{x}^k$ is similar to the data samples belonging to $\mathcal{DC}_i$, hence, it is necessary to update the number of points (s_i^k), the average ($\boldsymbol{\mu}_i^k$), and the variance ($(\sigma^2)_i^k$) of the cloud structure. Now, if $\mathbf{x}^k$ is considerably different from all existing *data clouds*, a new cloud is created. Thus, one of the conditions below may occur (Bezerra et al., 2016):

- **Condition 1**: $\mathbf{x}^k$ is not an outlier for at least one cluster; then update all the clusters for which this condition holds according to the following generalized versions of the *TEDA* equations:

$$\begin{aligned}
s_i^k &= s_i^{k-1} + 1 \\
\boldsymbol{\mu}_i^k &= \frac{s_i^k - 1}{s_i^k} \times \boldsymbol{\mu}_i^{k-1} + \frac{\mathbf{x}^k}{s_i^k} \\
(\sigma^2)_i^k &= \frac{s_i^k - 1}{s_i^k} \times (\sigma^2)_i^k + \frac{1}{s_i^k - 1} \times \left(\frac{2 \times ||\mathbf{x}^k - \boldsymbol{\mu}_i^k||}{\mathtt{D}} \right)^2 , \\
\xi_i(\mathbf{x}^k) &= \frac{1}{s_i^k} + \frac{2 \times (\boldsymbol{\mu}_i^k - \mathbf{x}^k)_T (\boldsymbol{\mu}_i^k - \mathbf{x}^k)}{s_i^k \times (\sigma^2)_i^k \times \mathtt{D}} \\
\zeta_i(\mathbf{x}^k) &= \frac{\xi_i(\mathbf{x}^k)}{2}
\end{aligned} \tag{5.9}$$

  in which $\mathtt{D}$ is the dimensionality of the data set.

- **Condition 2**: $\mathbf{x}^k$ is an outlier for all existing clusters; then create a new cluster with the following parameters:

$$\mathtt{NC} = \mathtt{NC} + 1, \quad s_{\mathtt{NC}}^k = 1, \quad \boldsymbol{\mu}_{\mathtt{NC}}^k = \mathbf{x}^k, \quad (\sigma^2)_{\mathtt{NC}}^k = 0 \, . \tag{5.10}$$

Similar to *TEDA*, the condition to define if a data point $\mathbf{x}^k$ is an outlier in relation to a *data cloud* $\mathcal{DC}_i$ is given by

$$\zeta_i(\mathbf{x}^k) > \frac{m^2 + 1}{2 \times s_i^k} \, . \tag{5.11}$$

However, the authors in Maia et al. (2020) argue that the condition expressed by Equation (5.11), with $m \geq 1$, will always be false when $s_i^k = 2$. That is, the second data point of the cluster $\mathcal{DC}_i$ will never be considered an outlier. This fact can be a problem especially when the first two data points are far away from each other. In practice, very large clusters can be generated and, as a consequence, dense regions can be improperly modeled in the data space. Thus, the authors in Maia et al. (2020) added a parameter r_0 to limit the variance of each cluster when $s_i^k = 2$ in order to prevent a cluster to grow indefinitely. This parameter modifies the outlier condition expressed by the Equation (5.11) as follows:

$$\zeta_2^i(\mathbf{x}_2) > \frac{m^2+1}{4} \quad \text{and} \quad (\sigma_2^i)^2 < r_0 \, . \tag{5.12}$$

They set the parameter r_0 to 0.001. In this work, we use that same value.

Figures 5.4 and 5.5 illustrate these two conditions by showing three data clouds, $\mathcal{DC}_1$, $\mathcal{DC}_2$ and $\mathcal{DC}_3$, and an input data sample $\mathbf{x}^k$ at the k-th time instant. In Figure 5.4, *TEDA*-Cloud calculates the normalized eccentricities of $\mathbf{x}^k$ ($\zeta_i(\mathbf{x}^k)$) in relation to all three *data clouds*. As $\mathbf{x}^k$ belongs to $\mathcal{DC}_2$ and $\mathcal{DC}_3$, these two *data clouds* are updated, while $\mathcal{DC}_1$ is kept unchanged. If the second condition is met as it is illustrated in Figure 5.5, a cloud $\mathcal{DC}_{\text{NC}+1}$ is created with the parameters presented in Equation (5.10). Note that $\mathbf{x}^k$ is a point considerably distant from all existing *data clouds*.

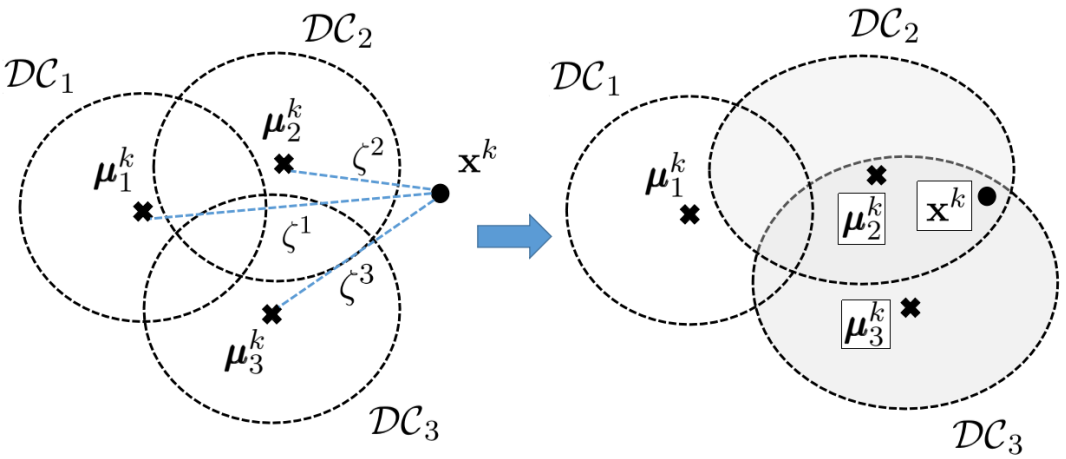


FIGURE 5.4: *Data clouds* updating: left illustration shows $\mathcal{DC}_1$, $\mathcal{DC}_2$, $\mathcal{DC}_3$ and a newly arrived data sample $\mathbf{x}^k$; right illustration shows the clouds after updating.

To adjust the *TEDA*-Cloud to the distributed model, we propose some adaptations as shown in Algorithm 2. The mechanisms presented in the algorithm are detailed below.

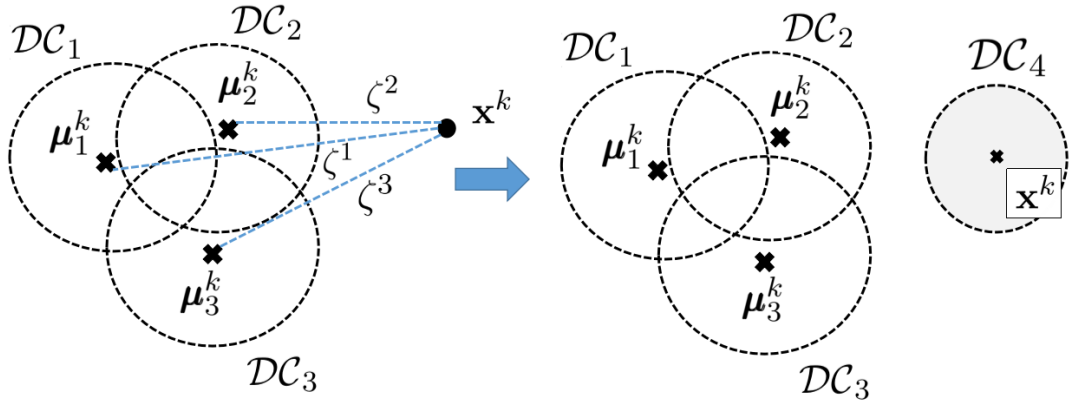


FIGURE 5.5: *Data clouds* creating: left illustration shows $\mathcal{DC}_1$, $\mathcal{DC}_2$, $\mathcal{DC}_3$ and a newly arrived data sample $\mathbf{x}^k$; right illustration shows the clouds after creation of $\mathcal{DC}_4$.

1. **Storage window of individuals:** As a recursive algorithm, *TEDA* does not need to store the data in memory, as only three main statistical resources are required for each *data cloud* $\mathcal{DC}_i$: s_i^k, $\boldsymbol{\mu}_i^k$ and $(\sigma^2)_i^k$ (Bezerra et al., 2016). However, we need to migrate an individual to the island. In our approach, we send the average individual considering a given number of best individuals of the cloud. Besides that, we have defined a restart mechanism if there is evidence that all *data clouds* are in the same search region. Therefore, we have established a maximum number w of individuals that can be saved in a *data cloud*. If the last individual $\mathbf{x}^k$ is not an outlier (line 12), it must be inserted in the *data cloud*. If the cloud is full, we define a sliding window scheme, in which the oldest individual is discarded, the window slides, and the individual $\mathbf{x}^k$ is inserted in the first position of the window (line 16).

2. **Restart mechanism:** Although the strategy of generating *data clouds* increases migration options, it is still possible that all clouds have similar individuals at some point in evolution. When this occurs, we have an indication that the islands are exploring the same search region, which can make migration unproductive. An excellent way to find out if this has occurred is when an incoming individual is not an outlier in any of the *data clouds*, i.e., for all *data clouds*, the condition in line 12 is true. To circumvent this problem, we propose a cloud restart mechanism when such a condition occurs (line 32). The idea is simple: we perform a perturbation in the *data cloud* $\mathcal{DC}_i$, according to the equation below:

$$\mathbf{x}_i = \mathcal{N}(\boldsymbol{\mu}_i, \sigma) + (\epsilon \times \mathbf{x}_{best} - \bar{\epsilon} \times \mathbf{x}_i)\,, \tag{5.13}$$

Algorithm 2 Function to perform clustering using *TEDA*-Cloud.

INPUT: $\mathbf{x}^k$: the best individual of the island at the k-th time instant; w: window size for storing individuals in *data clouds*; $\texttt{NC}_{max}$: maximum number of *data clouds*.
OUTPUT: $\mathcal{DC}_i, i = 1, 2, \ldots, \texttt{NC}$

```
1:  while new samples are arriving do
2:      if k == 1 then
3:          Set DC_1 parameters as defined in Equation (5.8)
4:      else
5:          flag_new_cluster ← true
6:          flag_restart ← true
7:          for i ← 1 to NC do
8:              if s_i^k == 2 then
9:                  outlier ← condition of Equation (5.12)
10:             else
11:                 outlier ← condition of Equation (5.11)
12:             if outlier == false then
13:                 if s_i^k < w then
14:                     Update DC_i according to Equation (5.9)
15:                 else
16:                     Slide the window of DC_i
17:                     Update DC_i according to Equation (5.9)
18:                 flag_new_cluster ← false
19:             else
20:                 flag_restart ← false
21:         if flag_new_cluster == true then
22:             if NC < MAX_NC then
23:                 Create a new cluster with the parameters of Equation (5.10)
24:             else
25:                 Randomly select a cloud DC_rand
26:                 if s_rand^k < w then
27:                     Update DC_rand according to Equation (5.9)
28:                 else
29:                     Slide the window of DC_rand
30:                     Update DC_rand according to Equation (5.9)
31:                                                        ▷ Restart mechanism
32:         if flag_restart == true then
33:             if NC > 1 then
34:                 for i ← 1 to NC do
35:                     Perform a perturbation on DC_i according to Equation (5.13)
36:                                                        ▷ Send an individual to the island
37:     Randomly select a cloud DC_rand among those where x^k is an outlier
38:     Send the DC_rand average individual, μ, to the island
```

in which ϵ and $\bar{\epsilon}$ are two uniform random numbers in the range $[0, 1]$, $\mathbf{x}_{best}$ is the best individual among the w individuals from $\mathcal{DC}_i$, $\mathbf{x}_i$ is the i-th individual among the w individuals from $\mathcal{DC}_i$, $\boldsymbol{\mu}_i$ and σ are the average and the standard deviation of $\mathcal{DC}_i$, respectively. Note that this perturbation occurs only when the number of clouds is greater than 1 (line 33).

3. **Number of clouds:** It is important to stress the goal of using the pool is to generate multiple migration options taking into account the migration history. Therefore, we are interested in generating a sufficient number of *data clouds* to provide some diversity. So, we have utilized a parameter ($\texttt{NC}_{max}$) to define the maximum number of *data clouds* (line

22). If the incoming individual is an outlier to all *data clouds* and $\texttt{NC} = \texttt{NC}_{max}$, this individual is added into an existing random cloud $\mathcal{DC}_{rand}$ (line 25), observing the issue of the storage window of individuals (lines 26–30). This strategy is also useful for generating diversity within the clouds.

Since clustering has been performed, we need to choose the individual that should migrate to the island. For this purpose, we select a cloud $\mathcal{DC}_{rand}$ randomly among those where $\mathbf{x}^k$ is an outlier. After that, we send the $\mathcal{DC}_{rand}$ average individual, $\boldsymbol{\mu}$, to the island. Thus, we seek to ensure that the migrated individual will generate diversity.

5.4 Final discussion

This clustering approach is proposed in order to keep a recent migration history. This history is used to generate clusters with similar and good-fitness individuals. However, this strategy needs to be efficient in identifying the changes in the parameters of the data distribution. The *TEDA*-based approach implements a clustering algorithm that evolves in an iterative, recursive, and autonomous way, which is very useful to deal with situations in which changes in data distributions are not usually known *a priori*. In this chapter, we adapted the idea to group similar individuals arriving at the pool. Finally, we use the *TEDA* to select migrated individuals that are sufficiently different from the requesting island.

Considering the DDMS and the adapted TEDA-Class, we can test 4 distributed strategies to recognize the contribution of each proposal: 1) `DDMS-TEDA`: it employs both diversity-driven migration strategy and adapted *TEDA*-Cloud; 2) `DDMS-BEST`: it uses the diversity-driven migration strategy to trigger the migrations, but it always sends the best individual; 3) `FIXED-TEDA`: it implements the adapted *TEDA*-Cloud to choose the migrated individual, but the migrations occur at a fixed interval, and; 4) `PROBA-TEDA`: it implements the adapted *TEDA*-Cloud to choose the migrated individual, but the migrations follow a probabilistic flow. In the next chapter, we compare these 4 migration strategies with the traditional migration strategies considering large-scale problems.

Chapter 6

Computational experiments

The computational experiments are divided into two parts. The first evaluates the performance of the proposed algorithms in terms of solution quality and running time. The second part of the experiments evaluates the timing of migrations and their impact on the convergence process. Besides that, we want to observe whether the proposed strategies are efficient in maintaining population diversity.

6.1 Test instances

A widely used large-scale optimization test suite provided in CEC'2013 is adopted to test the proposed approaches. The benchmark definitions can be found in Li et al. (2013). It should be noted that these test functions have been used in recent competitions of the IEEE Task Force on Large-Scale Global Optimization, such as the IEEE CEC'2021 Special Session and Competition on Large-Scale Global Optimization. Therefore, these are functions that adequately express the complexity of most real-world large-scale problems. These functions are classified into the following five groups:

1. **G1:** Fully-separable Functions ($f_1 - f_3$)
2. **G2:** Partially Additively Separable Functions with a separable subcomponent ($f_4 - f_7$)
3. **G3:** Partially Additively Separable Functions with no separable subcomponents ($f_8 - f_{11}$)
4. **G4:** Overlapping Functions ($f_{12} - f_{14}$)
5. **G5:** Non-separable Functions (f_{15})

6.2 Experimental design

We split the proposed strategies into 4 distributed algorithms to understand the contribution of each strategy. DDMS−TEDA employs both diversity-driven rules to trigger the migrations and *TEDA*-Cloud to choose the migrated individual. FIXED−TEDA and PROBA−TEDA implement *TEDA*-Cloud, but the migrations follow a constant flow. Finally, DDMS−BEST uses the diversity-driven rules, but it always sends the best individual. Table 6.1 summarizes these algorithms and lists the parameter settings of the distributed models. In all algorithms we use 4 islands (subpopulations) per subcomponent.

For each algorithm and test function, 30 independent runs were conducted with $\mathrm{NFE}^{max} = 3E+06$ and $\mathrm{D} = 1000$ (Li et al., 2013, Molina et al., 2018, Li et al., 2019, Chen et al., 2022). The control parameters of *TEDA*/*TEDA*-Class are: (1) the number of standard deviations distant from the mean for a data to be considered an outlier, m; (2) the maximum number of *data clouds*, NC^{max}; (3) the window size for storing individuals in *data clouds*, w. Table 6.2 lists the parameter settings.

With $m = 1$, we are not too rigorous when evaluating whether a point is an outlier in relation to a given data distribution. So, it is possible to generate multiple cluster options. Regarding NC^{max} and w parameters, we want to have multiple cluster options and a sufficient number of individuals within the clusters to generate diversity in migrations. We understand that NC^{max} and w equal to 10 is enough to promote this diversity. However, it may be useful to adjust these parameters in future experiments.

6.3 Implementation details

The algorithms were implemented in C++/MPI. The code was compiled using GNU Compiler Collection (GCC) version 7.5.0 with compiler optimization flag set to "-O3". MPI is a message-passing interface that allows a easy and efficient partitioning of the problem. It provides means of parallel communication among a distributed collection of processors (Karniadakis et al., 2003). The computational system is an Ubuntu 20.04 with 8 cores and 16GB of memory. In our implementation, we allocate a processor core to each island[1].

6.4 Performance of the proposed algorithms

A nonparametric statistical test called *Kruskal-Wallis* was used to determine if there are statistically significant differences between the groups. In case there is a difference, *Dunn* post-hoc

[1]The source codes are available for download at https://github.com/jeanto/ddms_lsgo.

TABLE 6.1: Parameter setting of distributed model for each strategy.

Algorithm	Parameter	Value	Description
Proposed Algorithms			
DDMS-TEDA	Migratory Topology	Pool	- Migrate individuals according to rules presented in Chapter 3. - Extract the average individual from one of the clouds where the best individual is an outlier according to TEDA-Cloud.
	Synchronization	Asynchronous	
	Migratory Policy	Mean individual substitutes random	
DDMS-BEST	Migratory Topology	Ring	- Migrate individuals according to rules presented in Chapter 3. - Send the best individual from the neighboring island.
	Synchronization	Asynchronous	
	Migratory Policy	Best individual substitutes random	
FIXED-TEDA	Migratory Topology	Pool	- Migrate individuals in a fixed interval. - Extract the average individual from one of the clouds where the best individual is an outlier according to TEDA-Cloud.
	Migratory Frequency	100	
	Synchronization	Synchronous	
	Migratory Policy	Mean individual substitutes random	
PROBA-TEDA	Migratory Topology	Pool	- Migrate individuals in a probabilistic interval. - Extract the average individual from one of the clouds where the best individual is an outlier according to TEDA-Cloud.
	Migratory Rate	0.05	
	Synchronization	Asynchronous	
	Migratory Policy	Mean individual substitutes random	
Traditional Algorithms			
FIXED-BEST	Migratory Topology	Ring	- Migrate individuals in a fixed interval. - Send the best individual from the neighboring island.
	Migratory Frequency	100	
	Synchronization	Synchronous	
	Migratory Policy	Best individual substitutes random	
PROBA-BEST	Migratory Topology	Ring	- Migrate individuals in a probabilistic interval. - Send the best individual from the neighboring island.
	Migratory Rate	0.05	
	Synchronization	Asynchronous	
	Migratory Policy	Best individual substitutes random	

TABLE 6.2: Parameter setting of *TEDA*/*TEDA*-Class.

Parameter	Value
m	1
NC^{max}	10
w	10

test with a *Bonferroni* adjustment was used to find out such differences (Dinno, 2015). We compare the proposed migration strategies (DDMS-TEDA, DDMS-BEST, FIXED-TEDA, and

PROBA-TEDA) against the traditional migration strategies (FIXED-BEST and PROBA-BEST). The idea is to show the count of wins, ties, and losses for each proposed strategy when compared to the two traditional strategies. That is, if a proposed strategy wins over both traditional strategies, then its count of wins is incremented. If a proposed strategy ties or losses to one of the traditional strategies, its count of ties or losses is incremented.

In the first round of experiments, we used three decomposition methods (DG, DG2, and RDG2), as presented in Chapter 3. The idea is to observe whether the proposed algorithms can achieve good results regardless of the decomposition method used. Initially, we implemented the algorithm *DE*/best/1 as an optimizer, with the scaling factor $F \in [0, 1]$ and the crossover rate $CR \in [0, 1]$ extracted from a uniform distribution. We defined the population size (NP) as equal to 100. The results are presented in Table 6.3, where the results of *Dunn's* test are summarized.

TABLE 6.3: Counts of wins, losses and ties of proposed strategies (DDMS-TEDA, DDMS-BEST, FIXED-TEDA, and PROBA-TEDA) *vs.* traditional strategies (FIXED-BEST and PROBA-BEST) according to significance of *Dunn's* test.

dec.	*proposed strategy*		f_1-f_3	f_4-f_7	f_8-f_{11}	f_{12}-f_{14}	f_{15}	Total
DG	**DDMS-TEDA**	# wins	0	**2**	**3**	**2**	**1**	**8**
		# losses	0	0	0	0	0	0
		# ties	3	2	1	1	0	7
	DDMS-BEST	# wins	0	**2**	**3**	**2**	**1**	**8**
		# losses	0	0	0	0	0	0
		# ties	3	2	1	1	0	7
	FIXED-TEDA	# wins	0	0	0	1	0	1
		# losses	0	0	0	1	0	1
		# ties	3	4	4	1	1	13
	PROBA-TEDA	# wins	0	0	0	0	0	0
		# losses	0	2	3	2	1	8
		# ties	3	2	1	1	0	7
DG2	**DDMS-TEDA**	# wins	0	**3**	**2**	**3**	**1**	**9**
		# losses	0	0	0	0	0	0
		# ties	3	1	2	0	0	6
	DDMS-BEST	# wins	0	**3**	**2**	**3**	**1**	**9**
		# losses	0	0	0	0	0	0
		# ties	3	1	2	0	0	6
	FIXED-TEDA	# wins	0	1	0	0	0	1
		# losses	0	0	0	0	0	0
		# ties	3	3	4	3	1	14
	PROBA-TEDA	# wins	0	0	0	0	0	0
		# losses	0	0	3	3	0	6
		# ties	3	4	1	0	1	9
RDG2	**DDMS-TEDA**	# wins	0	**4**	**4**	**3**	**1**	**12**
		# losses	0	0	0	0	0	0
		# ties	3	0	0	0	0	3
	DDMS-BEST	# wins	0	3	**4**	**3**	**1**	11
		# losses	0	0	0	0	0	0
		# ties	3	1	0	0	0	4
	FIXED-TEDA	# wins	0	1	0	0	0	1
		# losses	0	0	0	0	0	0
		# ties	3	3	4	3	1	14
	PROBA-TEDA	# wins	0	0	0	0	0	0
		# losses	0	0	0	0	0	0
		# ties	3	4	4	3	1	15

From a broader perspective, DDMS−TEDA and DDMS−BEST obtained the highest number of win counts among the proposed strategies, regardless of the decomposition method. These results indicate that the mechanism to determine the instant to carry out the migrations seems to have a strong positive impact on the results since the DDMS stands out. In other words, properly defining when to migrate can be more important than choosing which individual should be migrated.

Observe that FIXED−TEDA and PROBA−TEDA do not reach the same performance as DDMS−TEDA and DDMS−BEST. On the contrary, FIXED−TEDA and PROBA−TEDA tie with the traditional migration algorithms (FIXED−BEST and PROBA−BEST) in most scenarios. In fact, PROBA−TEDA is surpassed by FIXED−BEST and PROBA−BEST in most of the test functions when the decomposition algorithm is DG or DG2.

On the other hand, DDMS−TEDA and DDMS−BEST are better than traditional strategies in most test functions. Note that, in RDG2, DDMS−TEDA and DDMS−BEST overcome traditional strategies in 12 and 11 functions, respectively. This was the only difference observed between them.

Considering that DDMS−TEDA and DDMS−BEST obtained the best results, Table 6.4 shows the results of *Dunn's* test to observe if there is any statistical difference between them. The test demonstrates that there is no significant difference between the two strategies, regardless of the decomposition method used. They tied in all scenarios. These results suggest that it can be enough to send a highly-fit individual (the best one or the average of the n best) as long as the migrations occur at an appropriate time.

TABLE 6.4: Counts of wins, losses and ties of DDMS−TEDA *vs.* DDMS−BEST according to significance of *Dunn's* test.

proposed strategy	*dec.*		f_1-f_3	f_4-f_7	f_8-f_{11}	f_{12}-f_{14}	f_{15}	Total
DDMS−TEDA	DG	# wins	0	0	0	0	0	0
		# losses	0	0	0	0	0	0
		# ties	3	4	4	3	1	15
	DG2	# wins	0	0	0	0	0	0
		# losses	0	0	0	0	0	0
		# ties	3	4	4	3	1	15
	RDG2	# wins	0	0	0	0	0	0
		# losses	0	0	0	0	0	0
		# ties	3	4	4	3	1	15

Table 6.5 shows the results obtained by the proposed strategies as well as the results obtained by the traditional strategies. The results in this table are summarized as mean values for each function group. Columns f and *dec* are, respectively, the test function and the decomposition method. Note the results for the function group **G1** (Fully-separable functions $[f_1 - f_3]$) are not listed since there was no statistical difference between the methods in this group. The next columns contain the results obtained by the proposed and traditional strategies, in which *obj.* and t(s) columns contain, respectively, the mean values of the objective function, and the mean values of runtime in seconds.

TABLE 6.5: Results obtained by proposed and traditional migration strategies. The results presented in this table are the mean values of the objective function and the mean values of runtime in seconds for each group of functions and decomposition method.

			Proposed strategies								Traditional strategies			
			DDMS-TEDA		DDMS-BEST		FIXED-TEDA		PROBA-TEDA		FIXED-BEST		PROBA-BEST	
	f	*dec*	*obj.*	*t*(s)	*obj.*	*t*(s)	*obj.*	*t*(s)	*obj.*	*t*(s)	*obj.*	*t*(s)	*obj.*	*t*(s)
G2: Functions with a separable subcomponent	4	**DG**	2.28e+12	11	2.65e+12	11	**1.75e+12**	12	2.80e+12	11	1.96e+12	11	2.06e+12	11
		DG2	**4.57e+12**	08	4.58e+12	09	7.01e+12	08	1.06e+13	8	9.33e+12	08	9.34e+12	08
		RDG2	**4.03e+12**	10	4.04e+12	10	6.88e+12	10	1.03e+13	10	9.38e+12	10	9.64e+12	10
	5	DG	**3.95e+07**	07	3.96e+07	07	4.50e+07	07	4.80e+07	06	4.76e+07	06	4.87e+07	06
		DG2	3.54e+07	10	**3.51e+07**	10	4.35e+07	10	4.81e+07	09	4.65e+07	10	4.78e+07	10
		RDG2	3.25e+07	12	**3.24e+07**	12	4.27e+07	12	4.76e+07	12	4.63e+07	12	4.82e+07	12
	6	DG	1.07e+06	06	**1.07e+06**	06	1.07e+06	06	1.07e+06	06	1.07e+06	06	1.07e+06	06
		DG2	**1.06e+06**	15	1.06e+06	15	1.06e+06	15	1.06e+06	14	1.06e+06	14	1.06e+06	14
		RDG2	**1.06e+06**	18	1.06e+06	19	1.06e+06	18	1.06e+06	17	1.06e+06	17	1.06e+06	17
	7	**DG**	1.76e+11	07	**1.51e+11**	07	1.12e+12	06	6.34e+12	06	9.79e+11	06	4.72e+11	06
		DG2	**1.26e+12**	05	1.28e+12	05	5.18e+12	05	1.88e+13	05	6.33e+12	05	3.94e+12	05
		RDG2	5.16e+11	06	**4.38e+11**	06	3.74e+12	06	1.29e+13	05	4.78e+12	06	2.51e+12	06
G3: Functions with no separable subcomponents	8	**DG**	6.75e+16	20	**6.21e+16**	20	8.62e+16	20	1.39e+17	19	9.88e+16	20	1.24e+17	20
		DG2	1.53e+17	16	**1.50e+17**	16	2.50e+17	14	4.21e+17	14	1.68e+17	14	2.07e+17	15
		RDG2	1.46e+17	19	**1.35e+17**	19	2.55e+17	18	4.66e+17	17	2.59e+17	17	3.20e+17	18
	9	DG	**9.71e+08**	22	9.72e+08	22	1.37e+09	21	2.06e+09	20	1.40e+09	22	2.43e+09	21
		DG2	9.10e+08	19	**9.09e+08**	19	1.24e+09	19	1.84e+09	19	1.28e+09	18	1.55e+09	18
		RDG2	8.73e+08	23	**8.63e+08**	24	1.22e+09	23	1.80e+09	23	1.29e+09	21	1.52e+09	21
	10	DG	9.49e+07	20	**9.48e+07**	19	9.51e+07	19	9.54e+07	18	9.49e+07	19	9.50e+07	18
		DG2	9.44e+07	19	**9.44e+07**	19	9.45e+07	19	9.49e+07	18	9.45e+07	18	9.47e+07	19
		RDG2	9.43e+07	23	**9.43e+07**	24	9.46e+07	22	9.46e+07	23	9.46e+07	22	9.48e+07	23
	11	DG	8.44e+12	21	**8.08e+12**	21	1.70e+13	20	8.47e+13	20	1.93e+13	20	3.91e+13	20
		DG2	2.32e+11	17	**2.17e+11**	17	5.21e+11	17	2.30e+12	16	5.05e+11	16	8.39e+11	17
		RDG2	2.09e+11	21	**2.00e+1**1	21	4.78e+11	20	2.56e+12	21	5.16e+11	20	8.07e+11	20
G4: Overlapping Functions	12	**DG**	1.04e+05	0	2.25e+04	0	1.32e+08	01	3.79e+09	01	4.49e+04	0	**1.75e+04**	0
		DG2	**1.75e+11**	35	1.81e+11	36	1.21e+12	33	3.10e+12	33	1.27e+12	32	1.74e+12	32
		RDG2	1.33e+11	43	**1.33e+11**	44	1.21e+12	38	1.75e+12	39	1.31e+12	37	1.80e+12	38
	13	DG	**4.89e+12**	21	5.47e+12	21	5.03e+13	20	5.71e+14	19	5.53e+13	20	1.64e+14	19
		DG2	**3.34e+12**	18	4.05e+12	18	3.06e+13	18	4.99e+14	18	4.20e+13	17	1.40e+14	17
		RDG2	**1.30e+11**	39	1.34e+11	39	2.43e+12	37	1.42e+14	36	3.40e+12	35	8.64e+13	37
	14	DG	**1.64e+12**	21	1.77e+12	21	2.76e+12	20	7.78e+12	19	6.29e+12	19	1.52e+13	19
		DG2	1.01e+13	19	**9.94e+12**	19	2.75e+13	18	1.70e+14	17	2.94e+13	18	5.42e+13	17
		RDG2	6.13e+11	59	**6.12e+11**	59	1.05e+13	53	5.32e+14	53	1.36e+13	50	6.23e+14	51
G5: Non-SF	15	**DG**	2.00e+08	52	**1.95e+08**	51	5.93e+12	48	3.05e+14	49	1.60e+12	46	8.73e+13	45
		DG2	2.37e+08	42	**2.29e+08**	42	5.73e+12	39	2.98e+14	38	1.68e+12	38	1.32e+14	38
		RDG2	2.00e+08	51	**1.98e+08**	51	6.14e+12	46	2.88e+14	47	3.62e+12	46	1.72e+14	46

Regarding the objective function values, DDMS−TEDA and DDMS−BEST obtained the best mean values and alternated in the first position for the vast majority of cases. Overall, they were better in about 87% of the functions when *dec.* was equal to DG. When *dec.* was equal to DG2 or RDG2, DDMS−TEDA and DDMS−BEST get lower mean objective function values in 100% of the functions. It is therefore possible to state that the diversity-driven migration strategy has great potential to improve the performance of distributed evolutionary algorithms in terms of the mean value of the objective function.

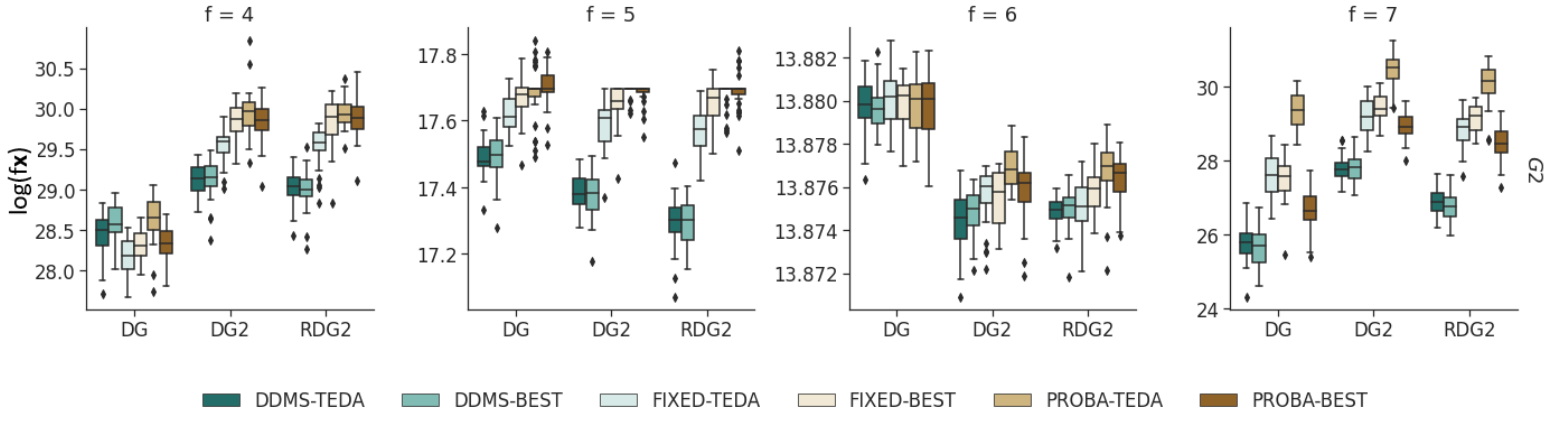
Although statistical tests have not shown any significant difference between DDMS−TEDA and DDMS−BEST, it is possible to observe that DDMS−BEST obtained lower mean values in most functions. One would think this is because sending the best individual during migrations promotes convergence. But, note that, in traditional strategies (FIXED−BEST and PROBA−BEST), even sending the best individual, the results are worse than the proposed strategies. Therefore, the results indicate that the moment of migration is the factor that can promote a significant positive impact on the performance of distributed evolutionary algorithms.

Regarding the runtime, the results do not indicate any significant increase when comparing the proposed and traditional strategies. The runtime values are similar mainly because the DDMS rules are implemented using TEDA framework concepts, where mean and standard deviation calculations are done recursively. The most visible difference is in f_{15}, where the DDMS−TEDA/DDMS−BEST spends an average of 6 minutes more than the traditional strategies. The main reason is that f_{15} is the only fully non-separable function, which impacts the calculation time of the DDMS rules since the 1000 variables are considered at once in the convergence and stagnation computation.

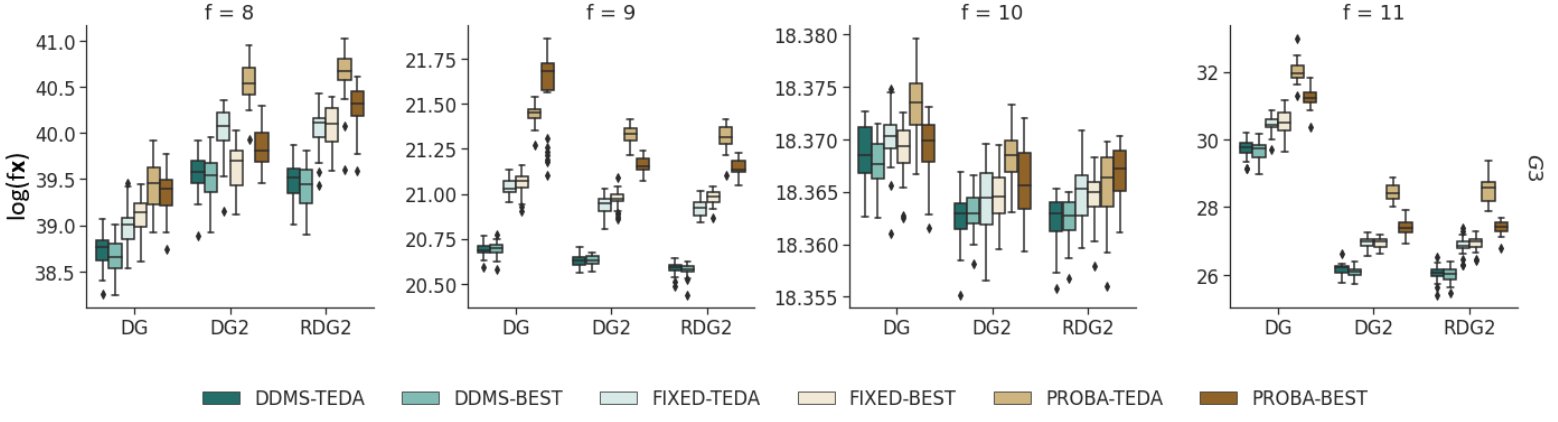
In addition to the results presented in the table above, Figure 6.1 shows boxplot graphs for the mean values of the objective functions of the solutions returned by the proposed and traditional strategies. The results are stratified by the decomposition method and group of functions. The logarithmic values are used for clarity.

From the results presented in the Table 6.5 and Figures 6.1a, 6.1b and 6.1c, it can be observed that both DDMS−TEDA and DDMS−BEST present similar performance in terms of the mean value of the objective function of the solutions returned by them.

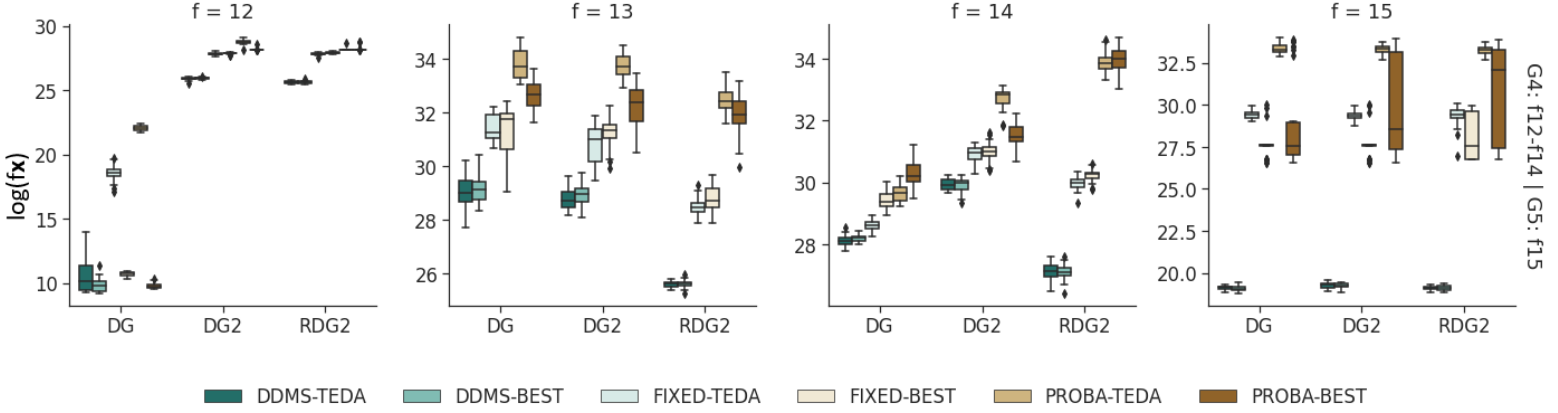
Figure 6.1a shows the results to **G2** (Partially Additively Separable Functions with a separable subcomponent). Note that DDMS−TEDA and DDMS−BEST clearly outperform the other strategies on functions f_5 and f_7, regardless of the decomposition algorithm. About **G3** (Partially Additively Separable Functions with no separable subcomponents), the same result appears in functions f_9 and f_{11}, as shown in Figure 6.1b. In the functions of **G4** (Overlapping Functions) and **G5** (Non-separable Function), DDMS−TEDA and DDMS−BEST stand out over the other strategies in functions f_{13}, f_{14}, and f_{15}, as illustrated in Figure 6.1c.



(A) Mean value of objective function obtained for functions with a separable subcomponent (**G2**).



(B) Mean value of objective function obtained for functions with no separable subcomponents (**G3**).



(C) Mean value of objective function obtained for functions with overlapping functions (**G4**) and Non-separable functions (**G5**).

FIGURE 6.1: Mean values of objective functions. The results are stratified by the decomposition method and group of functions.

In general, DDMS−TEDA and DDMS−BEST just don't outperform the other strategies at f_6 and f_{10} (Ackley Function). In its two-dimensional form, Ackley Function is characterized by a nearly flat outer region and a large hole at the center. Because of this characteristic, the algorithms present very similar performance. However, as shown in the Table 6.5, DDMS−TEDA and DDMS−BEST tend to reach lower mean values.

Regarding decomposition methods, DG performs better when dealing with functions with elliptical characteristics or subcomponents (f_4, f_7, f_8, and f_{12}). In the other functions, DG2 and RDG2 are more effective, especially RDG2 in functions f_5, f_9, f_{11}, f_{13}, f_{14}, and f_{15}.

It should be remembered that the optimizer used (*DE*/best/1) does not implement recent strategies to improve diversity in the optimization process. Therefore, migrations play this role. From these results, we can assume that the diversity-driven migration strategy has a strong positive impact on the results. In this context, we also want to observe whether these results are maintained if we use an evolutionary algorithm that implements strategies to maintain diversity, such as *SHADE*.

The control parameters of *SHADE* are: (1) population size, NP $= 100$; (2) external archive size, NPA $=$ NP; (3) historical memory size, H $=$ NP; (4) initial value of MCR and MF, $\mu_{CR}^{init} = 0.5$ and $\mu_{F}^{init} = 0.5$; (5) parameters to define p value for current-to-$p$best/1 mutation, $p_{min} = 2/\text{NP}$ and $p_{max} = 0.2$. These parameters follow the values of the original paper (Tanabe and Fukunaga, 2013).

To simplify and speed up the execution, this second round of experiments considers only the two best-proposed strategies (DDMS−TEDA and DDMS−BEST) compared to the traditional strategies (FIXED−BEST and PROBA−BEST). In addition, RDG2 is used as a decomposition method since it obtained the best average performance. Table 6.6 shows the results obtained for each group of functions.

Regarding the objective function values, DDMS−TEDA and DDMS−BEST obtained the best mean values in all cases. DDMS−BEST obtained the best results in 9 (f_5, $f_7 - f_{12}$, $f_{14} - f_{15}$) of the 12 analyzed functions, while DDMS−TEDA performed better in three functions (f_4, f_6, and f_{13}). Therefore, the diversity-driven migration strategy proved to be very promising even using an optimizer already known for its excellent performance. That is, DDMS can improve the performance of distributed evolutionary algorithms regardless of the optimizer(s) used within the islands. Again, these results suggest that a strategy to define a good moment to perform a migration can promote a significant difference in the performance of a DEA.

Figure 6.2 shows boxplot graphs for the mean values of the objective function of the solutions returned by the proposed and traditional strategies. The results are stratified by optimizer and group of functions. The logarithmic values are used for clarity. Optimizers are highlighted to compare the performance of traditional *DE*/best/1 against *SHADE*.

TABLE 6.6: **RDG2**: Results obtained by proposed and traditional migration strategies. The results presented in this table are the mean values of the objective function and the mean values of runtime in seconds for each group of functions.

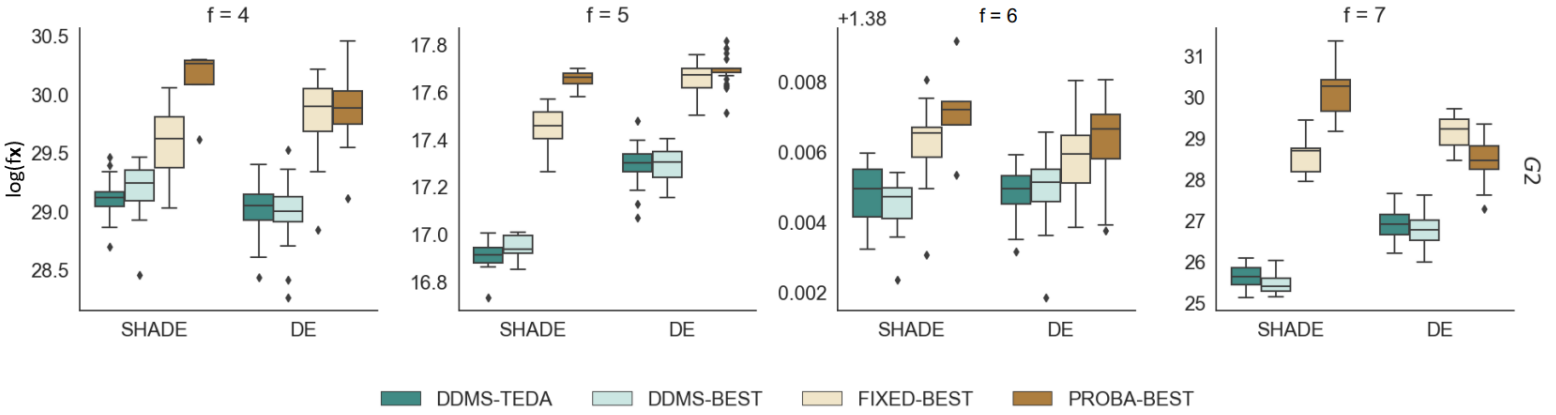
		Proposed strategies				Traditional strategies			
		DDMS−TEDA		DDMS−BEST		FIXED−BEST		PROBA−BEST	
	f	*obj.*	*t*	*obj.*	*t*	*obj.*	*t*	*obj.*	*t*
G2: Functions with a separable subcomponent	4	**4.03e+12**	10	4.04e+12	10	9.38e+12	10	9.64e+12	10
	5	3.25e+07	12	**3.24e+07**	12	4.63e+07	12	4.82e+07	12
	6	**1.06e+06**	18	1.06e+06	19	1.06e+06	17	1.06e+06	17
	7	5.16e+11	06	**4.38e+11**	06	4.78e+12	06	2.51e+12	06
G3: Functions with no separable subcomponents	8	1.46e+17	19	**1.35e+17**	19	2.59e+17	17	3.20e+17	18
	9	8.73e+08	23	**8.63e+08**	24	1.29e+09	21	1.52e+09	21
	10	9.43e+07	23	**9.43e+07**	24	9.46e+07	22	9.48e+07	23
	11	2.09e+11	21	**2.00e+1**1	21	5.16e+11	20	8.07e+11	20
G4: Overlapping Functions	12	1.33e+11	43	**1.33e+11**	44	1.31e+12	37	1.80e+12	38
	13	**1.30e+11**	39	1.34e+11	39	3.40e+12	35	8.64e+13	37
	14	6.13e+11	59	**6.12e+11**	59	1.36e+13	50	6.23e+14	51
G5: Non-SF	15	2.00e+08	51	**1.98e+08**	47	3.62e+12	46	1.72e+14	46

From the results, it can be observed that the proposed strategies that use *SHADE* present a better performance in most functions (f_5, f_7, f_9, $f_{11} - f_{15}$). In other functions (f_4, f_6, f_8, and f_{10}), both *SHADE* and *DE* present similar performance. It was already expected that the *SHADE* would perform better than the *DE*/best/1 since *SHADE* employs a success-history-based parameter adaptation that guides the selection of future control parameter values, which tends to improve the search efficiency.

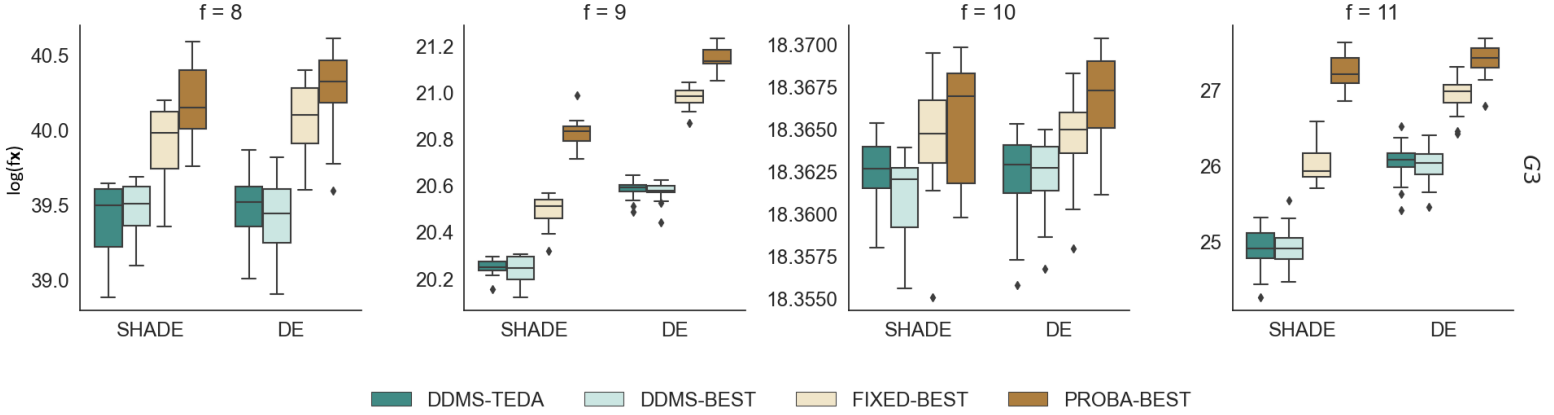
It is important to emphasize that the proposed strategies not only accompany but also enhance the improvement promoted by *SHADE*. Furthermore, it is interesting to note that in some cases *SHADE* is only significantly better than *DE* if it uses DDMS. This behavior can be shown in f_5, f_7, f_{11}, and f_{12}.

Focusing on functions in which *SHADE* was able to find the best solutions (i.e., f_5, f_7, f_9, $f_{11} - f_{15}$), it is possible to observe that there was no significant difference between DDMS−TEDA and DDMS−BEST. These results reinforce that the choice of the individual to be migrated is less decisive than the choice of the moment of migration.

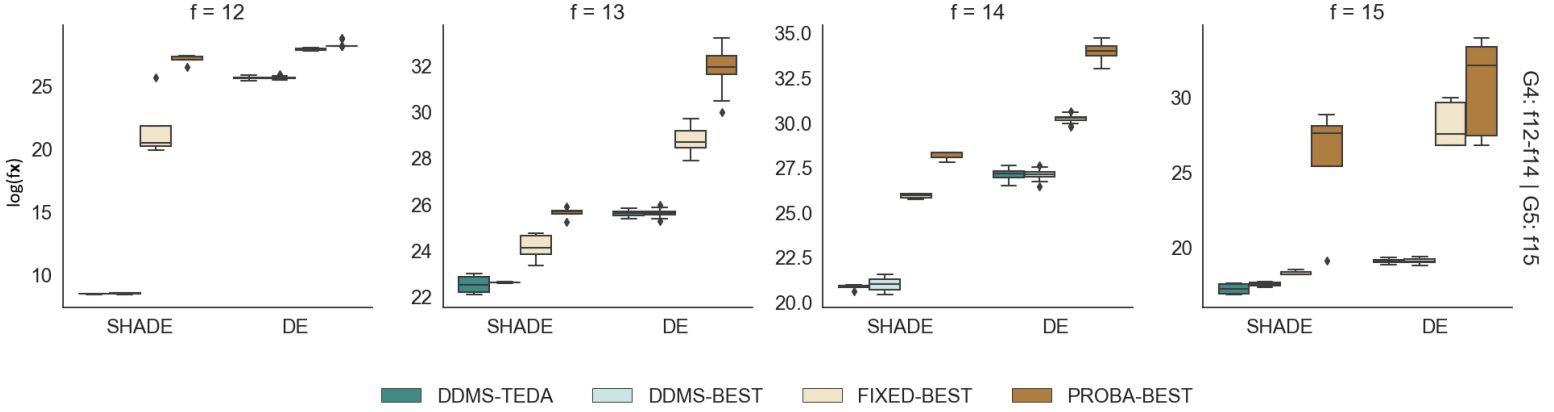
In these circumstances, it is possible to highlight that the TEDA-Cloud would add the possibility of having more options for migrating highly-fit individuals. That is, new possibilities can be explored, such as migrating more than one individual, choosing an individual who has the most potential to promote improvements on the requesting island, and creating clouds that favor exploration and clouds that favor exploitation, among others. The precise advantages of using TEDA-Cloud as part of the migration process go through the implementation of these proposals in future work.



(A) Mean value of objective function obtained for functions with a separable subcomponent (**G2**).



(B) Mean value of objective function obtained for functions with no separable subcomponents (**G3**).



(C) Mean value of objective function obtained for functions with overlapping functions (**G4**) and Non-separable functions (**G5**).

FIGURE 6.2: Mean values of objective functions. The results are stratified by the decomposition method and group of functions.

Figures 6.2a and 6.2b show the results to **G2** (Partially Additively Separable Functions with a separable subcomponent) and **G3** (Partially Additively Separable Functions with no separable subcomponents), respectively. The results are similar to those obtained in the first round of experiments. Figure 6.2c shows the results to **G4** (Overlapping Functions) and **G5** (Non-separable Function). Note that the main difference is at f_{12}, where the DDMS/*SHADE*/RDG2 combination promoted an extremely significant improvement when compared to the results obtained using *DE*/best/1.

6.5 Proposed mechanism's effectiveness

The results reinforce the choice of the proposed migration strategies, especially `DDMS-BEST` and `DDMS-TEDA`, rather than traditional strategies, allowing us to draw tentative conclusions regarding their performance. However, the precise advantages of using them as part of the distributed optimization framework remain to be investigated. The following sections will explore how the proposed ideas impact migration timing, convergence, and diversity. Taking these issues into account, we can assess whether migrations were efficient.

6.5.1 Effective moves after migration

We are also interested in evaluating when migration occurs and if it has been effective. If such a migration promoted any improvement, that is, the migrated individual is better than the best individual of the requesting island, we consider that an effective move has occurred.

We choose four functions according to the type of function, in which f_4 is from **G2**, f_9 is from **G3**, f_{12} is from **G4**, and f_{15} is from **G5**. Figures 6.3, 6.4, 6.5, and 6.6 illustrate the effective moves after migration in a typical run. The green bars mean that migration has occurred and it does not promote immediate improvement in the error value on the requesting island (migration without improvement). The red bars illustrate that migration has occurred and it promotes immediate improvement in the error value on the requesting island (migration with improvement). Some conclusions can be drawn as follows:

- Figures 6.3c, 6.3d, 6.4c, 6.4d, 6.5c, 6.5d, 6.6c, and 6.6d show that the migrations in traditional strategies are constant throughout evolution since they do not assess the loss of diversity on the islands. On the other hand, migrations in proposed strategies are more concentrated after half the run, which is when islands generally lose diversity, as shown in the Figures 6.3a, 6.3b, 6.4a, 6.4b, 6.5a, 6.5b, 6.6a, and 6.6b.
- `DDMS-TEDA` tries to apply the principle in which the migrated individual must not be too similar, but also not too different, to the subpopulation distribution of the requesting

island. In this strategy, the migrated individual must have a proper difference from the requesting island population while maintaining a historical quality standard. In practice, DDMS−TEDA has greater potential to temporarily restore the diversity of the island, which means less migration, as shown in Figures 6.3a, 6.4a, 6.5a, and 6.6a.

- Depending on the function, it is common for islands to converge to the same regions of the search space. According to DDMS, this fact tends to cause more migration due to the loss of diversity. The approach of sending the best individual helps to aggravate this decline in diversity. Therefore, DDMS−BEST has more migrations than DDMS−TEDA overall. In this sense, DDMS−TEDA seems to be the strategy that best suits the needs of the islands. This is because it seeks to maintain the historical backing of better individuals during migrations through the *TEDA*-Cloud. By clustering these individuals, it is possible to migrate an individual of good fitness and, at the same time, different from the requesting island population. However, this does not mean better mean values of objective function as it was shown in the previous section.

These results demonstrate the influence of the approach to define the proper moment of migration. In general, the average number of migrations of DDMS−BEST is significantly larger than other strategies, which suggests that the island subpopulations are trying to escape from a local minimum. This result shows two facts: i) the mechanism of determining when to migrate was effective, and ii) DDMS−BEST tends to be less effective than DDMS−TEDA in restoring island diversity.

6.5.2 Convergence

Figures 6.7 and 6.8 show the convergence plots for the island that obtained the lowest value of the objective function in a typical run using *SHADE* with RDG2. Note that, in general, DDMS−TEDA and DDMS−BEST converge faster. This is because our approach waits for the island to lose diversity to perform migrations. That is, DDMS allows islands to converge normally. On the other hand, traditional migration approaches perform periodic migrations, which can contribute to subpopulations being trapped in the same local minima. Then, after a while, the migrations stop taking effect.

In general, DDMS−BEST is a little more efficient than DDMS−TEDA in terms of convergence, as shown in the results of the previous section, although this difference was not statistically significant. From the results shown in Figures 6.7 and 6.8, it can be observed that sending the best individual allied to the DDMS tends to promote better results than the clustering mechanism of the DDMS−TEDA. It was somewhat predictable that this result would occur, given that the best individual sent by DDMS−BEST had already been recognized as a good solution. On the

other hand, the solution derived from TEDA-Cloud typically performs poorer in regards to the objective function as it is based on a kind of mutation.

Future works could explore novel methods of selecting the migrated individual as a means of enhancing convergence on DDMS−TEDA. However, in the upcoming section, we will observe that DDMS−TEDA has the potential to enhance diversity further.

6.5.3 Diversity

To be even more assertive about the overall performance of migration strategies, we are also interested in evaluating whether the migrations have promoted a gain in diversity. Along with previous results, information about diversity can support us to determine which strategies were able to drive convergence and meanwhile maintain or improve diversity on the islands.

Standard deviation is a frequently used statistic to assess the diversity of communities (Thukral et al., 2019). In our proposal to identify population convergence, we compute the standard deviation σ for each dimension j in each generation g (see Equation (4.2)). Having the σ_j^g matrix at the end of the run, we compute the average of σ_j according to the following equation:

$$\mu_{\sigma_j} = \frac{1}{\text{GEN}} \sum_{i=1}^{\text{GEN}} \sigma_{i,j} \,, \tag{6.1}$$

in which GEN is the number of generations at the end of the run. The result is a vector of averages of standard deviations. We generate a metric by computing the average of μ_σ. Considering the round of experiments using *SHADE* and RDG2, Table 6.7 summarizes this metric for each island, respectively. The numbers in bold highlight the strategies with the highest average standard deviation considering the 4 islands.

Considering the average ranks, DDMS−TEDA reaches the first position in all functions, followed by DDMS−BEST. Finally, the two traditional strategies were the worst ones. These results demonstrate apparently the influence of proper choice of the migrated individual so that diversity on the requesting island is improved. In fact, DDMS−BEST is the best method to guarantee smaller average errors because it imports the best individual. On the other hand, DDMS−TEDA is more effective in ensuring greater diversity mainly because it tries to import an average individual with good fitness. Overall, both approaches complement each other and perform similarly.

In summary, the results indicate the influence of the approach to define the proper moment of migration (DDMS). Also, TEDA proved its efficiency with the approach to improve diversity using clustering. In general, it is a promising idea to combine an approach to define the proper moment of migration together with an appropriate approach to promote diversity in distributed

evolutionary algorithms. For the latter case, the use of TEDA proved to be efficient due to its recursive and online characteristics.

TABLE 6.7: Average of μ_σ of the population.

Functions		Average of the μ_σ															
		DDMS-TEDA				DDMS-BEST				FIXED-BEST				PROBA-BEST			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
G1	f_1	**11.20**	**11.68**	**10.20**	**11.51**	10.15	10.28	10.54	10.81	1.61	1.44	1.43	1.12	1.91	1.43	1.12	1.61
	f_2	**11.23**	**14.11**	**7.83**	**12.18**	11.20	11.01	10.41	10.43	9.61	9.64	9.41	9.51	9.24	9.43	9.11	9.43
	f_3	**36.16**	**39.74**	**40.13**	**35.32**	36.32	34.72	33.31	36.87	21.07	22.07	18.61	19.21	11.21	11.81	11.33	10.31
G2	f_4	**0.55**	**0.54**	**0.51**	**0.48**	0.50	0.45	0.44	0.50	0.43	0.35	0.28	0.48	0.43	0.59	0.22	0.21
	f_5	**11.60**	**10.83**	**9.73**	**10.15**	7.51	8.50	8.12	8.31	5.81	6.23	5.41	6.32	3.35	3.81	3.44	3.46
	f_6	**13.96**	**15.91**	**16.41**	**16.05**	15.53	15.16	14.17	14.68	8.31	8.34	8.18	9.21	5.15	4.51	5.13	5.31
	f_7	**23.80**	**23.80**	**24.99**	**29.41**	21.12	21.43	22.53	22.10	7.56	7.44	7.62	7.71	16.12	16.21	16.48	16.21
	f_8	**11.12**	**14.41**	**7.87**	**13.88**	10.13	10.21	10.34	10.24	9.53	9.61	9.13	9.13	9.42	9.23	9.41	9.53
G3	f_9	**11.32**	**10.34**	**9.65**	**10.13**	7.51	8.01	8.53	8.41	5.81	6.93	5.93	6.12	3.54	3.81	3.55	3.12
	f_{10}	**35.75**	**34.01**	**38.63**	**38.31**	12.56	13.57	12.78	14.14	15.71	15.31	15.10	14.71	8.30	8.12	8.11	9.81
	f_{11}	**33.81**	**35.12**	**39.81**	**38.61**	26.91	28.50	25.91	28.86	18.41	18.89	17.41	15.90	10.71	11.61	10.93	11.90
G4	f_{12}	**32.85**	**38.61**	**37.21**	**38.61**	36.62	35.36	36.45	35.23	23.53	21.23	22.43	24.25	12.49	11.28	11.40	11.47
	f_{13}	**7.65**	**7.61**	**6.11**	**7.73**	6.91	6.21	6.01	6.32	4.21	3.55	3.17	3.86	2.13	2.32	2.10	2.51
	f_{14}	**5.32**	**5.21**	**6.45**	**5.82**	5.11	5.91	5.41	5.31	2.91	2.91	3.34	2.13	1.35	1.10	1.21	1.45
G5	f_{15}	**2.33**	**2.71**	**2.31**	**2.14**	2.14	2.11	2.13	2.14	1.23	1.89	1.99	1.54	0.64	0.94	1.01	0.99

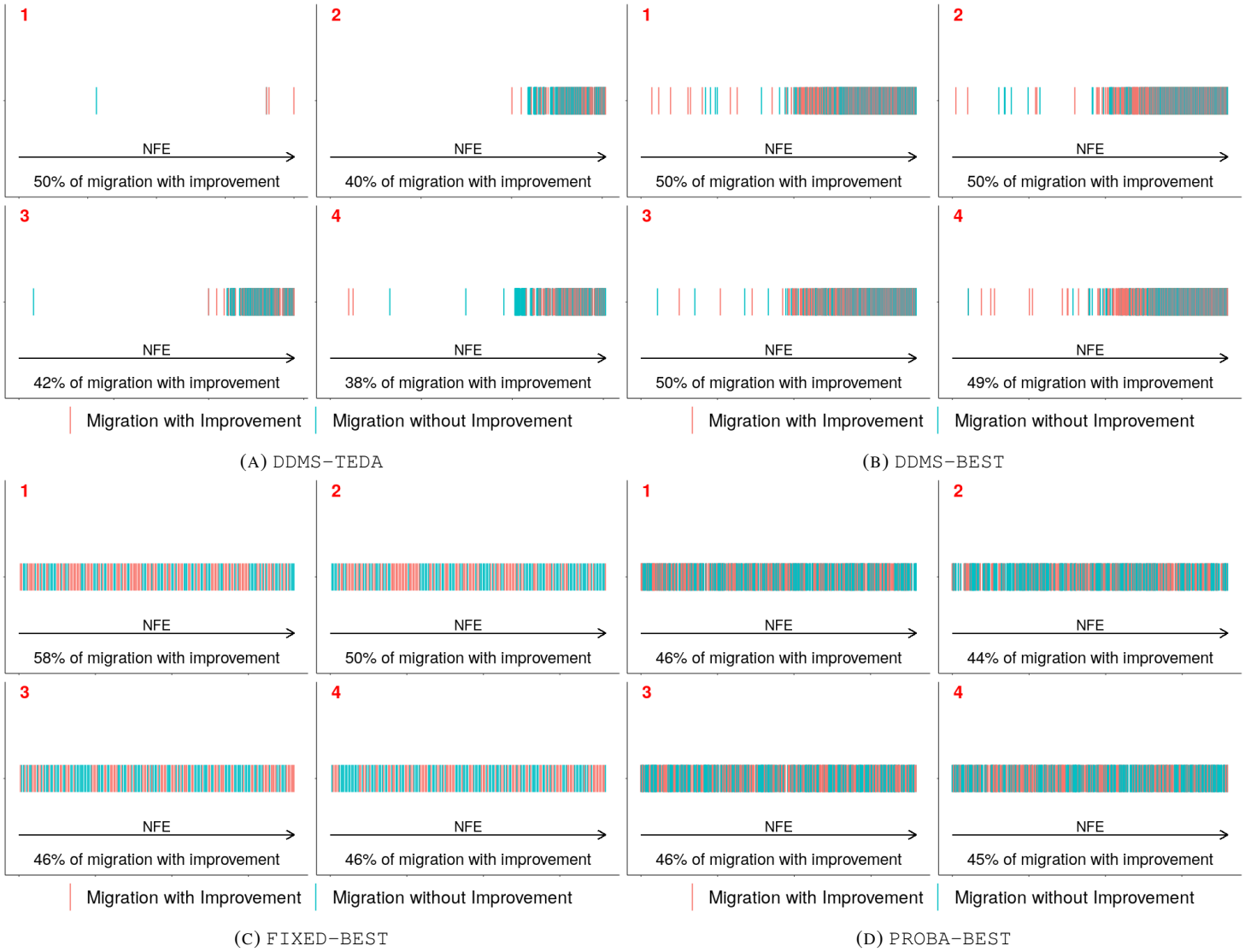
6.6 Results on moderate-scale optimization problems

The main objective of this work is to investigate the behavior of the proposed strategies in large-scale global optimization (LSGO) problems. Considering the promising performance, we extend the DDMS-TEDA analysis to problems with a moderate number of variables (50 and 100). In this case, it is not necessary to use a decomposition algorithm to perform the optimization.

For problems of this dimensionality, the literature generally uses the 30 test instances of the CEC'2014 Special Session and Competition on Single-Objective Real-Parameter Numerical Optimization (Liang et al., 2013, Awad et al., 2016, Zhang et al., 2019a, Wang et al., 2019, Tan et al., 2021, Halim et al., 2021, Wang et al., 2022).

Appendix A highlights some results of the DDMS-TEDA when compared with traditional migration strategies (FIXED-BEST and PROBA-BEST) and some prominent sequential evolutionary algorithms, namely, SHADE (Tanabe and Fukunaga, 2013), L-SHADE (Tanabe and Fukunaga, 2014), SHADE-ILS (Molina et al., 2018), L-SHADE-EpSin (Awad et al., 2016), FLDE (Tan et al., 2021), MSHCSA (Zhang et al., 2019a), and ADECSA (Wang et al., 2022).

In summary, DDMS-TEDA performed competitively against the most recent sequential *DE*-based and clonal algorithms, demonstrating that an efficient distributed proposal tends to improve the performance of prominent sequential evolutionary algorithms, even those not so recent, such as L-SHADE. In relation to traditional migration models in distributed algorithms, DDMS-TEDA showed as promising results as in large-scale problems. It had good performance especially for problems with 100 variables and hybrid/composition functions. The results presented in Appendix A demonstrate that the DDMS-TEDA is an effective strategy to promote better performance of evolutionary algorithms within a distributed model in smaller problems as well.

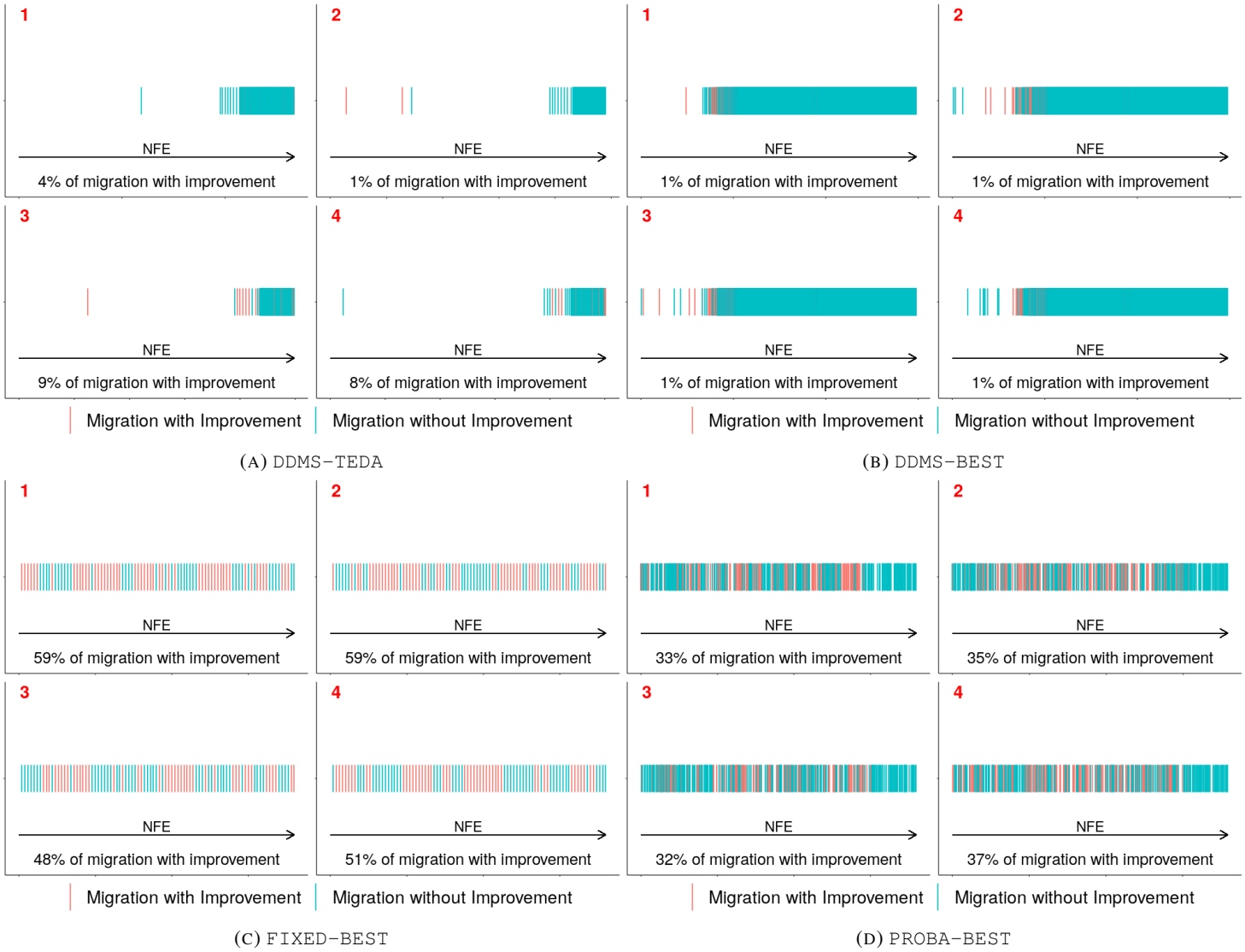


(A) DDMS−TEDA (B) DDMS−BEST

(C) FIXED−BEST (D) PROBA−BEST

FIGURE 6.3: f_4 (**G2**): Effective moves after migration in a typical run. The illustration shows the moment when a migration occurred and if such a migration promoted an improvement in the requesting island.

FIGURE 6.4: f_9 (**G3**): Effective moves after migration in a typical run. The illustration shows the moment when a migration occurred and if such a migration promoted an improvement in the requesting island.



(A) DDMS−TEDA (B) DDMS−BEST

(C) FIXED−BEST (D) PROBA−BEST

FIGURE 6.5: f_{12} (**G4**): Effective moves after migration in a typical run. The illustration shows the moment when a migration occurred and if such a migration promoted an improvement in the requesting island.

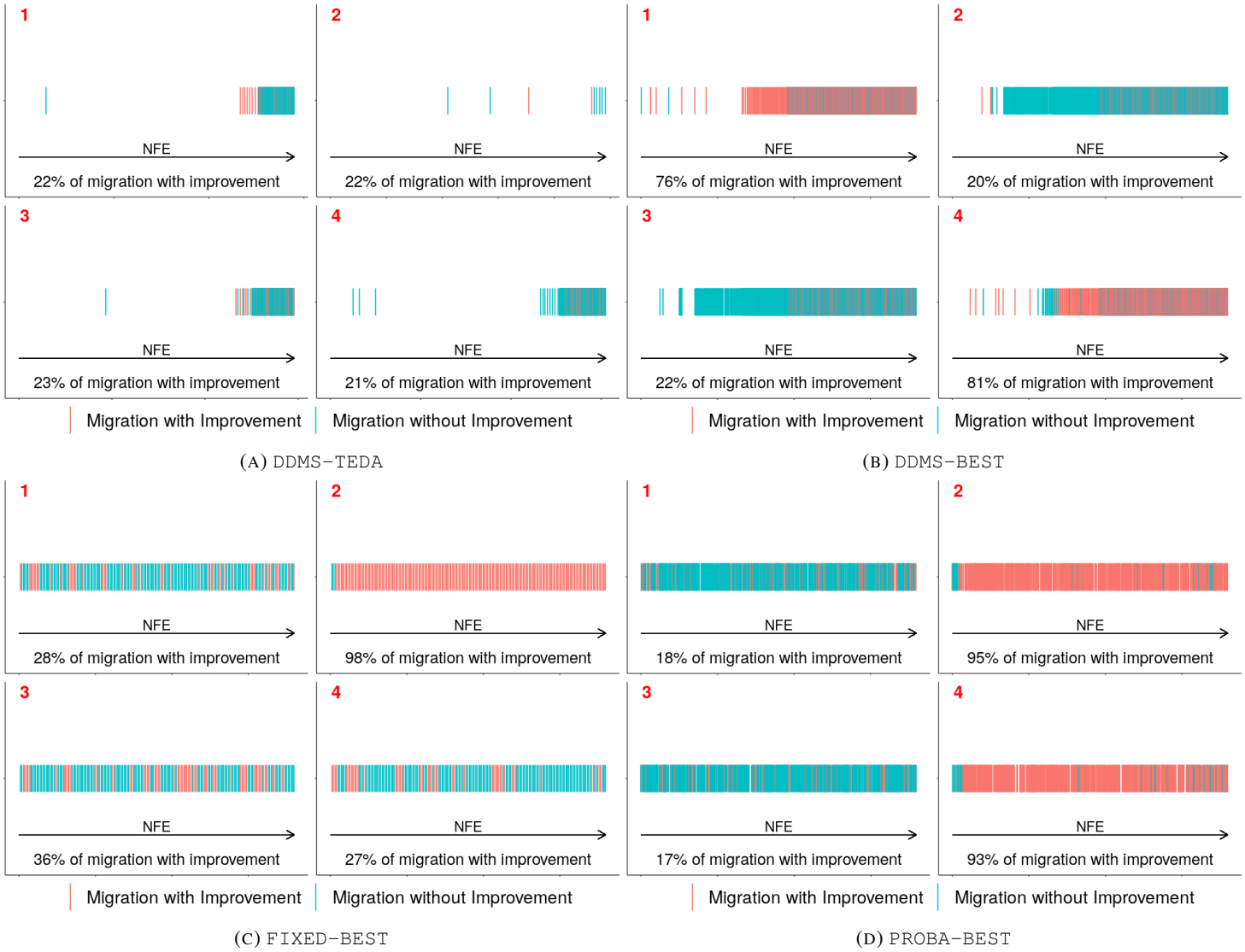
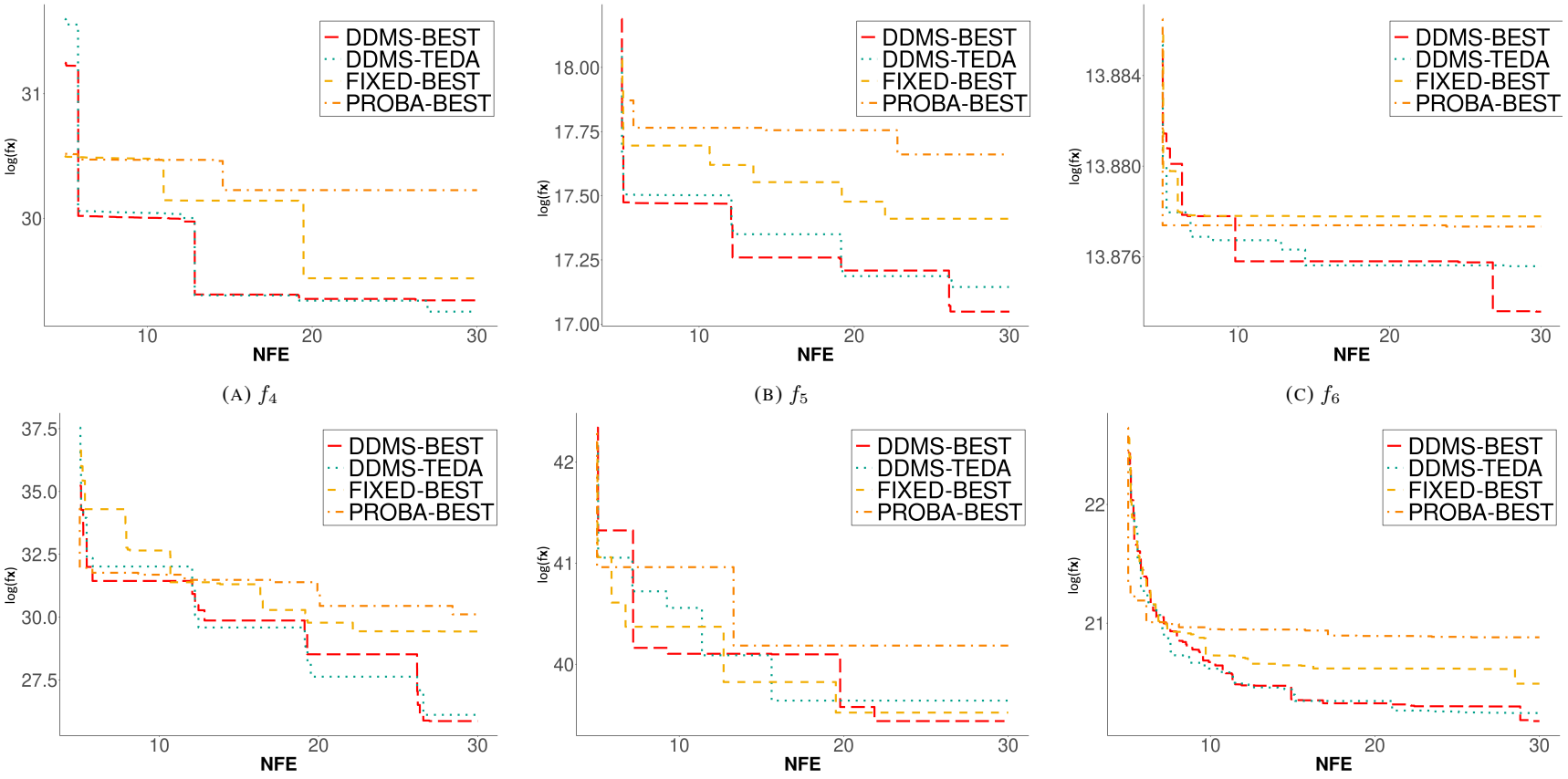


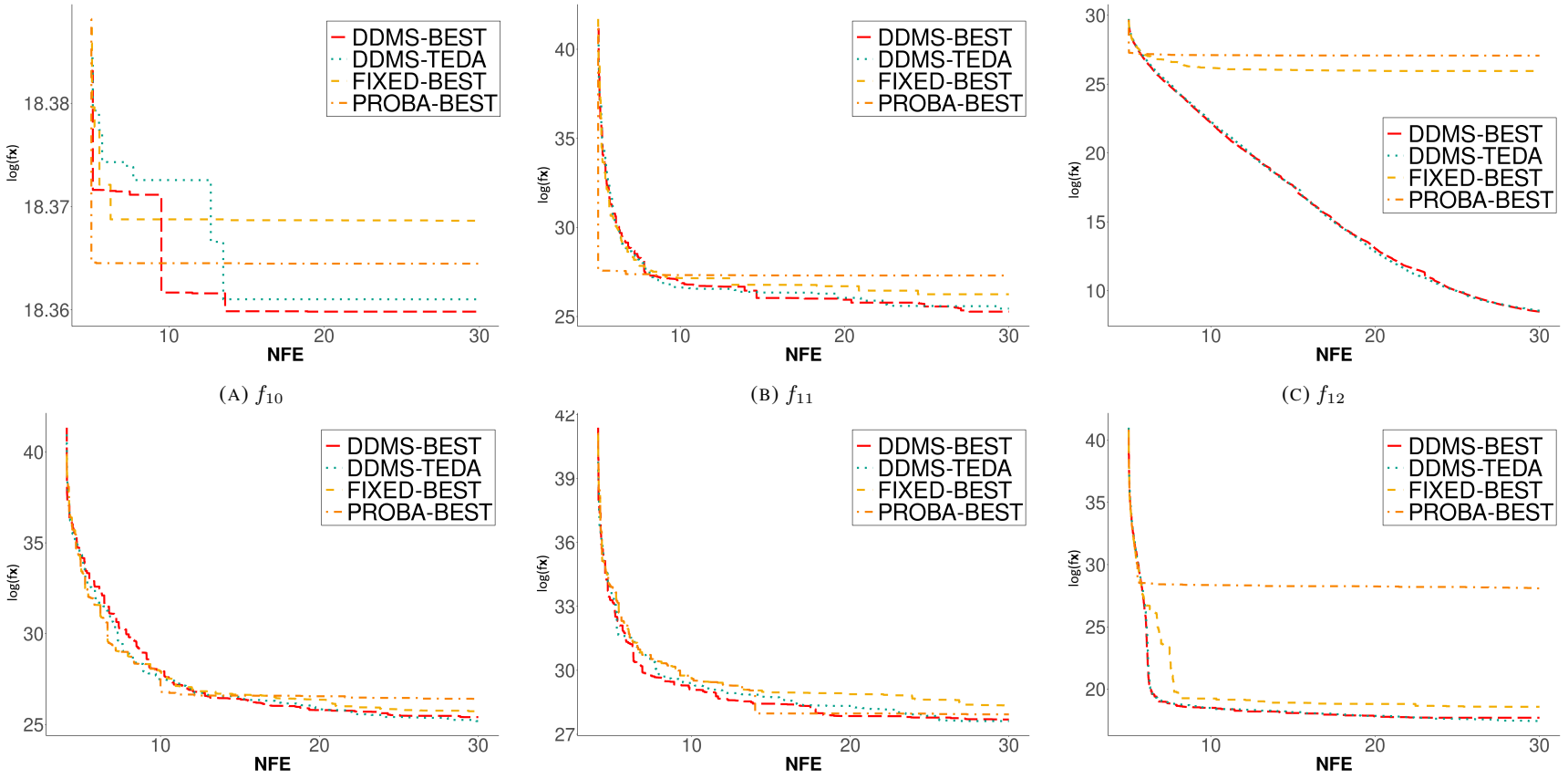
FIGURE 6.6: f_{15} (**G5**): Effective moves after migration in a typical run. The illustration shows the moment when a migration occurred and if such a migration promoted an improvement in the requesting island.



(A) f_4 (B) f_5 (C) f_6

(D) f_7 (E) f_8 (F) f_9

FIGURE 6.7: Convergence plots of the migration strategies for the best island for functions $f_4 - f_9$. Axes x (**NFE**) is multiplied by 10^5 for clarity.



(A) f_{10} (B) f_{11} (C) f_{12}

(D) f_{13} (E) f_{14} (F) f_{15}

FIGURE 6.8: Convergence plots of the migration strategies for the best island for functions $f_{10} - f_{15}$. Axes x (**NFE**) is multiplied by 10^5 for clarity.

Chapter 7

Final remarks

7.1 Overall conclusions

In the literature on distributed evolutionary algorithms, most of the approaches are proposed with migrating at fixed intervals or with a given probability. In this work, we recognize the importance of considering a more reliable estimation of the proper moment to perform the migration of an individual. This involves assessing the loss of diversity in each dimension and determining when to migrate. For that, we presented rules to assess recursively the loss of diversity in each dimension in order to obtain a trigger to perform timely migrations. We called this migration strategy the Diversity-driven Migration Strategy (`DDMS`).

Another important issue that has been little explored in the literature involves the definition of how to migrate an individual so that this migration promotes a productive improvement in the diversity of the requesting island. An efficient strategy would be to ensure the migrated individual is sufficiently different from the requesting island population and, at the same time, has good or moderate fitness so as not to harm convergence. Most studies focus on migrating the best or a random individual replacing the worst or a random individual on the requesting island.

In this work, we proposed two different strategies to define how the migration should be done. In the first strategy, called `DDMS-BEST`, the sending island sends its best individual to the requesting island to promote a diversity that does not negatively impact convergence. In the second strategy, we proposed an approach that aims to balance exploitation and exploration. Instead of creating a topology where there is direct communication between neighboring islands, migrations occur between the islands and a pool. After receiving the best individual from a given requesting island, this pool clusters it in a cloud together with other individuals which are similar by using the *TEDA*-Cloud framework. After clustering, an individual sufficiently different from the requesting island is extracted from one of the clouds and then the migration is

made. This approach seeks to maintain a recent history of good-fitness individuals while trying to generate diversity by sending individuals that are outliers when compared to the requesting island population. We combined this approach with DDMS and call it DDMS−TEDA.

The proposed strategies were compared against the usual approaches that consider periodic migrations in which a given individual on the requesting island is replaced by a copy of the best individual on the sending island. The experimental comparison was carried out on a test suite of 15 functions defined for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization. The strategies were tested with the dimension (D) set to 1000 variables.

The experimental comparison brings promising results for the proposed strategies. A deeper analysis of the comparisons' results told us that it is beneficial to use our Diversity-driven Migration Strategy (DDMS) along with a Cooperative Co-evolution Distributed Evolutionary model for solving large-scale optimization problems. Besides, DDMS performed better than traditional strategies (FIXED−BEST and PROBA−BEST) in the vast majority of functions considering two evolutionary algorithms (*DE/best/1* and SHADE) and three decomposition methods (DG, DG2, and RDG2). Regarding specifically diversity, DDMS−TEDA proved to be the most promising approach in most functions and scenarios. On convergence, DDMS−BEST is particularly more effective.

Although not the focus of this work, DDMS−TEDA was also tested for problems with 50 and 100 variables using the 30 test instances of the CEC'2014 Special Session and Competition on Single-Objective Real-Parameter Numerical Optimization. DDMS−TEDA were compared against the traditional migration approaches and prominent sequential evolutionary algorithms. Overall, DDMS−TEDA achieved excellent results, outperforming traditional migration models on more than half of the test problems in both scenarios. Furthermore, DDMS−TEDA was superior to sequential *SHADE*-based algorithms in the vast majority of functions.

These results should provide enough reason for adopting a proper diversity-based migration mechanism to reduce unnecessary and unproductive migrations. In particular, the proposed mechanisms have the potential to deal with high-dimensional, large-scale problems and composite landscapes that often refer to complex real-world problems.

7.2 Further works

Despite the promising results of this study, the research on diversity-based migration mechanisms should continue. We point out below some aspects that should be investigated in future works regarding large-scale optimization problems (LSOP).

The researches have shown the difficulties of metaheuristics in dealing with LSOP (Li et al., 2013, Omidvar et al., 2021b, Zhan et al., 2022). In fact, even though an EA implements mechanisms to balance exploration and exploitation, the curse of dimensionality is a problem that degrades the performance as the number of variables increases (Li et al., 2013, Chen et al., 2015). In this context, distributed evolutionary algorithms have attracted attention because of their ability to explore multiple search regions by dividing the population into subgroups (Ge et al., 2017, Jia et al., 2018). Distributed models have shown very promisingly in many works (Ge et al., 2017, Meng et al., 2017, Wang et al., 2019, Li et al., 2022).

Another common strategy for improving the performance of EAs for LSOPs is to decompose the problem into several low-dimensional subproblems. The algorithms that use this strategy are called Cooperative Co-evolution Evolutionary Algorithms (CCEAs) (Li et al., 2013, Chen et al., 2015). However, maintaining diversity remains an open challenge for both traditional metaheuristics (Omidvar et al., 2021b) and CCEAs (Omidvar et al., 2021a) when applied to LSOPs, even when the algorithm is distributed due to the significant impact of the migration strategy (Wang et al., 2019). This work presents a new approach that combines a CCEA with a distributed model that seeks to make the migration process efficient and well-adapted to the diversity needs of the islands. However, some improvements and adaptations need to be addressed in future works.

- The results showed that migration often does not produce an immediate result in terms of both convergence and diversity. When this occurs, it is innate in our mechanism for assessing the loss of diversity that a new migration takes place right afterward. As a consequence, it was common to observe many migrations occurring in consecutive generations. In future work, we wish to develop a scheme to measure the historical contribution, in terms of convergence and diversity, of the migrations that have occurred on a given island. The goal is to determine whether or not migration is a better option at that very moment. To this end, new actions must be implemented other than just migration.

- Some situations can happen with some recurrence in distributed models. First, a couple of islands may be exploring the same search region. Second, the islands' contribution to convergence and diversity can differ greatly. Third, computational resources are limited and can change during the evolutionary process. Fourth, migrations may be ineffective in terms of convergence and diversity for several generations in a row. Therefore, we wish to use some statistics to implement actions such as mergence and split of islands, inclusion or removal of islands according to the computational resources available, and migration schemes better suited to the needs of the islands and the recent migration history.

- In the approach to determine when to migrate in Chapter 4, section 4.2, we proposed a rule that follows the premise that convergence and stagnation are much less likely to

occur at the beginning of the evolutionary process. Although the proposed approach in Equation (4.11) can identify situations of diversification, this premise may not apply to all cases. It may happen that, as the algorithm evolves, the probability of applying diversification increases regardless of the real need of the island. Therefore, an improvement to be considered is the addition of a condition that is not tied to this time parameter. In future works, we want to introduce a metric to assess the proportion of times rule 3 is triggered in the past n generations to obtain a clearer indication of the real need for the island.

- In `DDMS-TEDA`, new possibilities can be explored, such as migrating more than one individual, choosing an individual who has the most potential to promote improvements on the island, and creating clouds that favor exploration and clouds that favor exploitation, among others. Also, we hope to make a sensitivity analysis in relation to the parameters of the *TEDA*-Class algorithm. For example, we want to assess whether the maximum number of *data clouds* ($\texttt{NC}^{max}$) directly impacts the results.

- In this work, `DDMS-BEST` uses a ring topology. Future works can also investigate the behavior of strategies in different distributed topologies.

- In future experiments, it is necessary to evaluate the scalability of the proposed approaches.

- In future versions, PCA (Principal Component Analysis) and similar approaches can be used for dimensionality reduction in subproblems with a large number of variables.

Appendix A

Computational experiments on CEC'2014 special session and competition

Thirty test instances, proposed in the CEC'2014 special session and competition on single-objective real-parameter numerical optimization, were used to study the performance of the DDMS-TEDA in scenarios with moderate number of variables. A detailed description of these test instances can be found in Liang et al. (2013). In this benchmark, $f_1 - f_3$ are unimodal functions (U), $f_4 - f_{16}$ are simple multimodal functions (SM), $f_{17} - f_{22}$ are hybrid functions (H), and $f_{23} - f_{30}$ are composition functions (C).

For each algorithm and each test function, 30 independent runs were conducted with NFE^{max} = $10000 \times$ D evaluations as the termination criterion, in which D is the number of dimensions. When the objective function gap between the best found solution and the optimal one is 10^{-8} or less, the error (score) is treated as 0, and the algorithm terminates. In these experiments, we define two scenarios, with D $= 50$ and D $= 100$.

We have used *Kruskal-Wallis* test to determine if there are statistically significant differences between the groups. In case there is a difference, *Dunn* post-hoc test with a *Bonferroni* adjustment was used to find out such differences (Dinno, 2015).

A.1 DDMS-TEDA *vs.* Traditional migration strategies

First, we compare the DDMS-TEDA against the traditional migration strategies (FIXED-BEST and PROBA-BEST). The control parameters of DDMS-TEDA, FIXED-BEST, and

PROBA-BEST are the same as presented in the section 6.2 (Experimental design) of Chapter 6. The idea is to show the count of wins, ties, and losses for the proposed strategy when compared to the two traditional strategies. That is, if the DDMS-TEDA wins over both traditional strategies, then its count of wins is incremented. If the DDMS-TEDA ties or losses to one of the traditional strategies, its count of ties or losses is incremented. In all strategies, we use L-SHADE as the optimizer. These results are shown in Tables A.1 and A.2.

The results reveal that the count of wins of DDMS-TEDA is higher especially for hybrid (H) and composition (C) functions. Such a result indicates DDMS-TEDA can deal with complex landscapes that usually resemble real-world problems. Observe DDMS-TEDA performs better than traditional migration strategies (FIXED-BEST and PROBA-BEST) in more than half of the test problems, and the count of wins is larger than the count of losses in both scenarios.

TABLE A.1: Counts of wins, losses and ties of DDMS-TEDA *vs.* FIXED-BEST according to significance of *Dunn's* test.

	D = 50					D = 100				
DDMS-TEDA	U	SM	H	C	Tot.	U	SM	H	C	Tot.
# wins	0	7	5	3	15	0	6	4	7	17
# losses	0	5	1	4	10	0	7	2	0	9
# ties	3	1	0	1	5	3	0	0	1	4

TABLE A.2: Counts of wins, losses and ties of DDMS-TEDA *vs.* PROBA-BEST according to significance of *Dunn's* test.

	D = 50					D = 100				
DDMS-TEDA	U	SM	H	C	Tot.	U	SM	H	C	Tot.
# wins	0	7	5	3	15	0	6	5	8	19
# losses	0	5	1	3	9	0	6	0	0	6
# ties	3	1	0	2	6	3	1	1	0	5

The error values ($f(\mathbf{x}) - f(\mathbf{x}^*)$) are given in Tables A.3 and A.4 for 50 and 100 variables, respectively. The numbers in parentheses represent the standard error and the p-value of each algorithm. We added a bold highlight to the strategies that obtained the lowest average error for each function. With D $=$ 50, DDMS-TEDA achieves the first position in 19 functions and outperforms other ones in 15 functions. For D $=$ 100, DDMS-TEDA achieves the first position in 22 functions and outperforms other ones in 17 functions.

All strategies had good results in optimizing the unimodal functions (f_1–f_3). In simple multimodal functions (f_4–f_{16}), we can see a balance between DDMS-TEDA and traditional strategies. In hybrid and composition functions (f_{17}–f_{30}), DDMS-TEDA had good performance, especially in composition functions for D = 100. In fact, Tables A.3 and A.4 show that DDMS-TEDA is more favorable for solving problems with hybrid and composition functions.

TABLE A.3: Average error on D = 50. Numbers in bold highlight represent the strategies with the lowest average error. Numbers in parentheses represent the standard error and the *p*-value, respectively.

Fun.	DDMS-TEDA	FIXED-BEST	PROBA-BEST
f_1	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**
f_2	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**
f_3	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**
f_4	1.36e+01 (±1.10e+00)(−)	**1.39e−01 (±1.58e−01)(2.11e−15)**	7.08e−01 (±4.40e−02)(2.41e−05)
f_5	2.02e+01 (±3.09e−03)(−)	**2.00e+01 (±5.18e−03)(0.00e+00)**	**2.00e+01 (±3.97e−03)(0.00e+00)**
f_6	1.27e−05 (±9.58e−07)(−)	**0.00e−00 (±1.78e−10)(1.08e−08)**	**0.00e+00 (±0.00e+00)(1.08e−08)**
f_7	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**
f_8	1.04e−08 (±7.86e−10)(−)	**0.00e+00 (±0.00e+00)(2.19e−14)**	**0.00e+00 (±0.00e+00)(2.19e−14)**
f_9	**1.42e+01 (±1.63e−01)(−)**	3.05e+01 (±1.88e−01)(6.99e−15)	2.97e+01 (±2.56e−01)(6.65e−11)
f_{10}	**2.85e−01 (±3.09e−03)(−)**	1.11e+01 (±4.74e−02)(5.14e−12)	9.99e+00 (±5.01e−03)(5.78e−06)
f_{11}	**2.68e+03 (±2.62e+01)(−)**	4.25e+03 (±1.83e+01)(2.19e−14)	4.97e+03 (±1.19e+01)(0.00e+00)
f_{12}	1.60e−01 (±1.13e−03)(−)	**5.16e−02 (±2.13e−03)(0.00e+00)**	6.77e−02 (±1.42e−03)(2.46e−11)
f_{13}	**1.17e−01 (±1.17e−03)(−)**	1.23e−01 (±1.15e−03)(4.03e−03)	1.21e−01 (±1.76e−03)(7.85e−03)
f_{14}	**2.59e−01 (±4.76e−04)(−)**	2.99e−01 (±1.52e−03)(3.02e−10)	2.96e−01 (±1.47e−03)(2.53e−07)
f_{15}	**4.69e+00 (±2.45e−02)(−)**	4.85e+00 (±1.74e−02)(1.48e−02)	4.84e+00 (±2.76e−02)(1.95e−03)
f_{16}	**1.57e+01 (±2.44e−02)(−)**	1.77e+01 (±3.65e−02)(1.11e−15)	1.80e+01 (±3.46e−02)(0.00e+00)
f_{17}	**3.48e+02 (±8.77e+00)(−)**	5.94e+02 (±1.37e+01)(5.14e−12)	4.75e+02 (±1.25e+01)(4.62e−05)
f_{18}	**9.00e+01 (±4.33e−01)(−)**	9.47e+01 (±6.23e−01)(8.81e−04)	1.03e+02 (±7.96e−01)(6.23e−13)
f_{19}	8.57e+00 (±9.36e−02)(−)	6.50e+00 (±2.43e−01)(2.41e−05)	**6.29e+00 (±1.62e−01)(6.23e−13)**
f_{20}	**8.06e+00 (±1.18e−01)(−)**	9.43e+00 (±1.16e−01)(2.06e−04)	1.08e+01 (±2.01e−01)(1.74e−12)
f_{21}	**3.57e+02 (±4.74e+00)(−)**	3.72e+02 (±4.47e+00)(1.20e−02)	3.88e+02 (±5.43e+00)(8.83e−05)
f_{22}	**1.36e+02 (±1.84e+00)(−)**	1.78e+02 (±1.23e+00)(2.03e−08)	1.83e+02 (±2.48e+00)(2.03e−08)
f_{23}	**1.24e+02 (±0.00e+00)(−)**	**1.24e+02 (±0.00e+00)(1.00e+00)**	**1.24e+02 (±0.00e+00)(1.00e+00)**
f_{24}	**2.70e+02 (±8.38e−02)(−)**	2.71e+02 (±1.74e−01)(4.03e−03)	2.71e+02 (±1.56e−01)(4.03e−03)
f_{25}	2.00e+02 (±2.19e−04)(−)	**2.00e+02 (±0.00e+00)(1.09e−02)**	2.00e+02 (±1.98e−04)(3.69e−02)
f_{26}	1.00e+02 (±6.96e−04)(−)	**1.00e+02 (±7.55e−04)(4.00e−04)**	1.00e+02 (±1.48e−03)(6.50e−01)
f_{27}	4.00e+02 (±3.21e−05)(−)	**4.00e+02 (±4.97e−05)(1.00e−04)**	4.00e+02 (±5.88e−05)(1.00e−04)
f_{28}	5.73e+02 (±4.29e−01)(−)	**5.71e+02 (±9.31e−01)(8.81e−04)**	**5.71e+02 (±9.19e−01)(3.73e−04)**
f_{29}	**7.48e+02 (±3.58e+00)(−)**	7.69e+02 (±2.19e+00)(4.03e−03)	7.90e+02 (±3.11e+00)(2.03e−08)
f_{30}	**6.23e+02 (±1.44e+00)(−)**	6.69e+02 (±2.83e+00)(4.05e−11)	6.38e+02 (±1.44e+00)(2.50e−03)

TABLE A.4: Average error on D = 100. Numbers in bold highlight represent the strategies with the lowest average error. Numbers in parentheses represent the standard error and the *p*-value, respectively.

Fun.	DDMS-TEDA	FIXED-BEST	PROBA-BEST
f_1	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**
f_2	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**
f_3	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**
f_4	**9.34e+01 (±2.50e+00)(−)**	**8.70e+01 (±1.38e+00)(5.00e−06)**	**8.68e+01 (±4.57e+00)(5.45e−05)**
f_5	2.04e+01 (±1.33e−02)(−)	**2.00e+01 (±1.69e−04)(2.46e−13)**	**2.00e+01 (±3.46e−04)(6.98e−06)**
f_6	**7.01e+00 (±6.11e−02)(−)**	4.59e+01 (±2.91e−01)(0.00e+00)	4.58e+01 (±6.20e−01)(0.00e+00)
f_7	**0.00e+00 (±0.00e+00)(−)**	2.53e−03 (±2.73e−04)(5.00e−06)	5.39e−03 (±6.62e−04)(0.00e+00)
f_8	2.86e−01 (±1.33e−02)(−)	**0.00e+00 (±0.00e+00)(0.00e+00)**	4.52e−01 (±1.17e−01)(1.88e−01)
f_9	**4.08e+01 (±4.02e−01)(−)**	1.05e+02 (±8.85e−01)(0.00e+00)	1.09e+02 (±1.70e+00)(0.00e+00)
f_{10}	5.52e+01 (±2.94e−01)(−)	**1.23e−02 (±9.50e−04)(0.00e+00)**	8.47e−02 (±1.08e−02)(1.39e−14)
f_{11}	9.62e+03 (±2.30e+01)(−)	**8.57e+03 (±2.63e+01)(5.14e−11)**	9.46e+03 (±7.27e+01)(1.23e−01)
f_{12}	3.33e−01 (±2.06e−03)(−)	**1.04e−01 (±1.47e−03)(1.39e−14)**	1.41e−01 (±2.35e−03)(1.29e−04)
f_{13}	**2.11e−01 (±5.07e−04)(−)**	3.42e−01 (±2.86e−03)(0.00e+00)	3.19e−01 (±3.15e−03)(7.33e−05)
f_{14}	**1.05e−01 (±6.07e−04)(−)**	1.14e−01 (±6.00e−04)(1.29e−04)	1.28e−01 (±6.27e−04)(0.00e+00)
f_{15}	**1.23e+01 (±1.61e−01)(−)**	2.03e+01 (±2.99e−01)(4.05e−08)	2.18e+01 (±4.41e−01)(1.66e−09)
f_{16}	3.76e+01 (±4.11e−02)(−)	**3.59e+01 (±5.49e−02)(4.27e−13)**	3.65e+01 (±1.24e−01)(1.29e−04)
f_{17}	**2.23e+03 (±2.01e+01)(−)**	3.78e+03 (±2.94e+01)(1.41e−10)	3.95e+03 (±7.92e+01)(6.66e−16)
f_{18}	**2.15e+02 (±6.50e−01)(−)**	2.35e+02 (±1.91e+00)(1.66e−09)	2.32e+02 (±1.47e+00)(1.46e−07)
f_{19}	1.04e+02 (±2.61e−01)(−)	9.06e+01 (±1.31e+00)(1.33e−15)	1.05e+02 (±1.22e+00)(4.82e−01)
f_{20}	**3.42e+01 (±2.54e−01)(−)**	1.54e+02 (±2.19e+00)(0.00e+00)	1.38e+02 (±2.50e+00)(1.62e−06)
f_{21}	**6.60e+02 (±9.26e+00)(−)**	1.81e+03 (±2.83e+01)(3.67e−12)	1.77e+03 (±2.58e+01)(3.11e−11)
f_{22}	8.88e+02 (±1.22e+01)(−)	**8.49e+02 (±8.26e+00)(8.75e−04)**	1.10e+03 (±2.27e+01)(1.29e−04)
f_{23}	**1.23e+02 (±0.00e+00)(−)**	**1.23e+02 (±0.00e+00)(1.00e+00)**	1.40e+02 (±3.44e+00)(1.30e−12)
f_{24}	**3.71e+02 (±1.02e−01)(−)**	3.76e+02 (±3.50e−01)(3.96e−05)	3.79e+02 (±3.97e−01)(1.41e−13)
f_{25}	**2.01e+02 (±8.84e−04)(−)**	2.01e+02 (±1.19e−03)(5.09e−04)	2.01e+02 (±1.27e−03)(1.67e−12)
f_{26}	**1.00e+02 (±1.32e−03)(−)**	1.00e+02 (±3.62e−03)(0.00e+00)	1.00e+02 (±3.47e−03)(2.16e−12)
f_{27}	**4.00e+02 (±3.75e−05)(−)**	4.00e+02 (±2.17e−04)(0.00e+00)	4.00e+02 (±2.56e−04)(0.00e+00)
f_{28}	**8.31e+02 (±1.85e+00)(−)**	1.03e+03 (±4.92e+00)(0.00e+00)	1.04e+03 (±9.38e+00)(0.00e+00)
f_{29}	**7.13e+02 (±1.36e+00)(−)**	8.19e+02 (±7.51e+00)(7.33e−05)	9.83e+02 (±2.41e+01)(0.00e+00)
f_{30}	**2.15e+03 (±1.26e+01)(−)**	3.12e+03 (±1.96e+01)(4.55e−15)	3.31e+03 (±3.45e+01)(0.00e+00)

A.2 DDMS-TEDA *vs.* Sequential EAs

Tables A.5, A.6, and A.7 show the count of wins, ties, and losses for DDMS-TEDA when compared to Sequential EAs (SHADE, L-SHADE and SHADE-ILS) in scenarios with 50 and 100 variables. *SHADE-ILS* (Molina et al., 2018) is a more recent improved version of *SHADE* that incorporates an Iterative Local Search that combines the exploration power of a contemporary DE algorithm with the exploitation ability of several local search methods.

The parameter settings of *SHADE*, *L-SHADE* and *SHADE-ILS* are the same as those used in their original papers (Tanabe and Fukunaga, 2013, 2014, Molina et al., 2018).

The results show that DDMS-TEDA is significantly better than Sequential EAs in more than two-thirds of the functions, and the count of wins is much larger than the count of losses in both scenarios.

TABLE A.5: Counts of wins, losses and ties of DDMS-TEDA *vs.* SHADE according to significance of *Dunn's* test.

	D = 50					D = 100				
DDMS-TEDA	U	SM	H	C	Tot.	U	SM	H	C	Tot.
# wins	0	9	6	7	22	0	13	5	4	22
# losses	0	2	0	0	2	0	0	1	1	2
# ties	3	2	0	1	6	3	0	0	3	6

TABLE A.6: Counts of wins, losses and ties of DDMS-TEDA *vs.* L-SHADE according to significance of *Dunn's* test.

	D = 50					D = 100				
DDMS-TEDA	U	SM	H	C	Tot.	U	SM	H	C	Tot.
# wins	0	9	6	7	22	0	12	5	4	21
# losses	0	4	0	0	4	0	1	1	1	3
# ties	3	0	0	1	4	3	0	0	3	6

TABLE A.7: Counts of wins, losses and ties of DDMS-TEDA *vs.* SHADE-ILS according to significance of *Dunn's* test.

	D = 50					D = 100				
DDMS-TEDA	U	SM	H	C	Tot.	U	SM	H	C	Tot.
# wins	0	8	2	5	15	0	12	5	4	21
# losses	0	3	2	1	6	0	1	1	1	3
# ties	3	1	3	2	9	3	0	0	3	6

In general, distributed approaches tend to be more efficient than sequential ones, mainly because inter-island communication can help the population within the island to escape from local minimums.

The error values ($f(\mathbf{x}) - f(\mathbf{x}^*)$) are given in Tables A.8 and A.9 for 50 and 100 variables, respectively. The numbers in parentheses represent the standard error and the p-value of each algorithm, respectively. We added a bold highlight to the strategies that obtained the lowest average error for each function.

Table A.8 shows that the DDMS−TEDA achieves the first position in 24 functions when D = 50. Besides, it outperforms other algorithms in 19 functions. Table A.9 shows the corresponding results on D = 100. For the average performance, it achieves the first position in 27 functions and outperforms the other ones in 21 functions.

Putting the type of test function into perspective, none of the strategies had difficulty in optimizing the unimodal instances (f_1–f_3). In other functions (f_4–f_{30}), DDMS−TEDA had good performance mainly when the number of variables is equal to 100.

Considering SHADE−ILS was the best one among sequential methods, we can assume that DDMS−TEDA tends to improve the performance of EAs since it was able to overcome SHADE−ILS in most functions even running L−SHADE on the islands.

TABLE A.8: Average error on D = 50: DDMS−TEDA *vs.* Sequential EAs. Numbers in bold highlight represent the strategies with the lowest average error. Numbers in parentheses represent the standard error and the *p*-value, respectively.

Fun.	DDMS−TEDA	SHADE	L−SHADE	SHADE−ILS
f_1	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**
f_2	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**
f_3	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**
f_4	**1.36e+01 (±1.10e+00)(−)**	3.04e+01 (±4.18e−02)(4.18e−03)	2.97e+01 (±3.11e−02)(2.49e−03)	2.19e+01 (±8.12e−02)(1.03e−03)
f_5	2.02e+01 (±3.09e−03)(−)	2.03e+01 (±2.46e−03)(2.18e−01)	**2.01e+01 (±9.56e−04)(1.65e−04)**	**2.01e+01 (±1.14e−03)(8.18e−05)**
f_6	**1.27e−05 (±9.58e−07)(−)**	1.29e+01 (±1.33e−07)(1.65e−04)	3.20e−01 (±3.83e−02)(1.65e−04)	1.28e+01 (±1.07e−01)(1.41e−07)
f_7	**0.00e+00 (±0.00e+00)(−)**	5.91e−03 (±9.21e−08)(3.18e−14)	5.42e−03 (±0.00e+00)(2.19e−14)	5.42e−03 (±0.00e+00)(2.19e−14)
f_8	1.04e−08 (±7.86e−10)(−)	**0.00e+00 (±0.00e+00)(9.92e−13)**	**0.00e+00 (±0.00e+00)(2.19e−14)**	**0.00e+00 (±0.00e+00)(1.98e−14)**
f_9	**1.42e+01 (±1.63e−01)(−)**	3.17e+01 (±3.37e−01)(0.00e+00)	3.11e+01 (±4.98e−01)(0.00e+00)	3.13e+01 (±2.18e−01)(0.00e+00)
f_{10}	2.85e−01 (±3.09e−03)(−)	5.67e−03 (±3.40e−01)(5.14e−03)	**1.86e−03 (±2.86e−01)(1.65e−04)**	1.91e−03 (±2.11e−01)(1.73e−04)
f_{11}	**2.68e+03 (±2.62e+01)(−)**	4.22e+03 (±2.13e+01)(7.31e−08)	3.51e+03 (±5.04e+01)(1.65e−04)	3.66e+03 (±1.98e+01)(8.21e−04)
f_{12}	**1.60e−01 (±1.13e−03)(−)**	**1.59e−01 (±1.74e−03)(7.91e−01)**	**1.58e−01 (±2.52e−03)(5.69e−01)**	**1.58e−01 (±1.99e−03)(4.78e−01)**
f_{13}	**1.17e−01 (±1.17e−03)(−)**	2.92e−01 (±1.76e−03)(0.00e+00)	2.90e−01 (±2.07e−03)(0.00e+00)	2.91e−01 (±2.06e−03)(0.00e+00)
f_{14}	**2.59e−01 (±4.76e−04)(−)**	3.58e−01 (±6.79e−03)(0.00e+00)	3.52e−01 (±2.04e−03)(0.00e+00)	3.19e−01 (±1.75e−03)(0.00e+00)
f_{15}	**4.69e+00 (±2.45e−02)(−)**	6.66e+00 (±4.59e−02)(0.00e+00)	6.58e+00 (±4.88e−02)(0.00e+00)	5.40e+00 (±3.58e−02)(0.00e+00)
f_{16}	**1.57e+01 (±2.44e−02)(−)**	1.74e+01 (±2.49e−02)(1.22e−06)	1.74e+01 (±8.90e−02)(1.27e−06)	1.69e+01 (±3.86e−02)(7.35e−05)
f_{17}	**3.48e+02 (±8.77e+00)(−)**	1.45e+03 (±2.07e+01)(0.00e+00)	1.44e+03 (±2.20e+01)(0.00e+00)	9.92e+02 (±1.86e+01)(0.00e+00)
f_{18}	**9.00e+01 (±4.33e−01)(−)**	1.62e+02 (±2.23e+00)(0.00e+00)	1.60e+02 (±9.66e−01)(0.00e+00)	**9.01e+01 (±5.66e−01)(2.17e−01)**
f_{19}	8.57e+00 (±9.36e−02)(−)	1.03e+01 (±1.55e−01)(1.16e−04)	1.01e+01 (±4.23e−02)(1.65e−04)	**7.99e+00 (±1.29e−01)(1.24e−02)**
f_{20}	**8.06e+00 (±1.18e−01)(−)**	5.19e+01 (±1.26e+00)(0.00e+00)	5.17e+01 (±1.96e−01)(0.00e+00)	**7.93e+00 (±1.36e−01)(1.87e−01)**
f_{21}	**3.57e+02 (±4.74e+00)(−)**	1.49e+03 (±2.01e+02)(0.00e+00)	1.51e+03 (±8.62e+00)(0.00e+00)	**3.61e+02 (±6.21e+00)(3.72e−01)**
f_{22}	1.36e+02 (±1.84e+00)(−)	3.49e+02 (±5.33e+00)(0.00e+00)	3.52e+02 (±6.44e+00)(0.00e+00)	**1.15e+02 (±6.42e+00)(2.45e−03)**
f_{23}	**1.24e+02 (±0.00e+00)(−)**	**1.24e+02 (±0.00e+00)(1.00e+00)**	**1.24e+02 (±0.00e+00)(1.00e+00)**	**1.24e+02 (±0.00e+00)(1.00e+00)**
f_{24}	**2.70e+02 (±8.38e−02)(−)**	2.80e+02 (±1.61e−01)(0.00e+00)	2.80e+02 (±1.35e−01)(0.00e+00)	2.75e+02 (±3.62e−02)(0.00e+00)
f_{25}	**2.00e+02 (±2.19e−04)(−)**	2.00e+02 (±1.16e−03)(8.54e−09)	2.00e+02 (±1.37e−03)(1.30e−09)	2.00e+02 (±1.04e−03)(4.66e−08)
f_{26}	**1.00e+02 (±6.96e−04)(−)**	1.00e+02 (±2.99e−03)(9.26e−12)	1.00e+02 (±1.52e−03)(7.71e−13)	1.00e+02 (±1.29e−03)(6.62e−12)
f_{27}	**4.00e+02 (±3.21e−05)(−)**	4.00e+02 (±1.40e−04)(6.64e−05)	4.00e+02 (±4.79e−05)(1.00e−04)	4.00e+02 (±2.31e−05)(1.23e−02)
f_{28}	5.73e+02 (±4.29e−01)(−)	6.06e+02 (±1.29e+00)(9.39e−08)	6.06e+02 (±1.02e+00)(1.11e−07)	**5.58e+02 (±1.28e+00)(9.14e−03)**
f_{29}	**7.48e+02 (±3.58e+00)(−)**	8.86e+02 (±8.10e+00)(0.00e+00)	8.94e+02 (±6.27e+00)(0.00e+00)	7.94e+02 (±2.42e+00)(0.00e+00)
f_{30}	**6.23e+02 (±1.44e+00)(−)**	9.62e+02 (±1.16e+01)(0.00e+00)	9.60e+02 (±4.77e+00)(0.00e+00)	6.45e+02 (±3.80e+00)(0.00e+00)

TABLE A.9: Average error on D = 100: DDMS−TEDA *vs.* Sequential EAs. Numbers in bold highlight represent the strategies with the lowest average error. Numbers in parentheses represent the standard error and the *p*-value, respectively.

Fun.	DDMS−TEDA	SHADE	L−SHADE	SHADE−ILS
f_1	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	7.93e−06 (±0.00e+00)(1.39e−14)
f_2	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	0.00e+00 (±0.00e+00)(1.00e+00)
f_3	**0.00e+00 (±0.00e+00)(−)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	**0.00e+00 (±0.00e+00)(1.00e+00)**	0.00e+00 (±0.00e+00)(1.00e+00)
f_4	**9.34e+01 (±2.50e+00)(−)**	1.09e+02 (±2.65e+00)(6.95e−04)	1.09e+02 (±1.28e+00)(6.69e−04)	1.06e+02 (±2.94e+00)(9.11e−03)
f_5	**2.04e+01 (±1.33e−02)(−)**	2.05e+01 (±4.67e−03)(4.25e−04)	2.05e+01 (±1.85e−04)(3.85e−04)	2.05e+01 (±4.25e−03)(4.01e−04)
f_6	**7.01e+00 (±6.11e−02)(−)**	3.90e+01 (±1.53e−01)(1.01e−04)	3.91e+01 (±2.77e−01)(1.29e−04)	3.87e+01 (±1.89e−01)(9.92e−03)
f_7	**0.00e+00 (±0.00e+00)(−)**	5.48e−03 (±3.80e−04)(0.00e+00)	5.29e−03 (±2.26e−04)(0.00e+00)	5.25e−03 (±3.59e−04)(0.00e+00)
f_8	2.86e−01 (±1.33e−02)(−)	2.97e−08 (±5.86e−09)(3.57e−04)	2.59e−08 (±0.00e+00)(6.32e−05)	**2.50e−08 (±5.38e−09)(3.36e−06)**
f_9	**4.08e+01 (±4.02e−01)(−)**	9.43e+01 (±4.86e−01)(2.01e−04)	9.43e+01 (±7.49e−01)(1.29e−04)	9.40e+01 (±4.42e−01)(7.65e−03)
f_{10}	**5.52e+01 (±2.94e−01)(−)**	7.78e−01 (±3.66e−02)(8.14e−03)	7.92e−01 (±9.44e−04)(1.29e−04)	7.68e−01 (±3.15e−02)(5.13e−03)
f_{11}	**9.62e+03 (±2.30e+01)(−)**	1.14e+04 (±1.62e+01)(7.12e−07)	1.14e+04 (±2.96e+01)(7.47e−07)	1.13e+04 (±1.43e+01)(6.34e−07)
f_{12}	**3.33e−01 (±2.06e−03)(−)**	3.70e−01 (±1.68e−03)(1.76e−04)	3.70e−01 (±1.48e−03)(1.29e−04)	3.69e−01 (±1.16e−03)(9.33e−03)
f_{13}	**2.11e−01 (±5.07e−04)(−)**	3.48e−01 (±2.01e−03)(0.00e+00)	3.47e−01 (±2.45e−03)(0.00e+00)	3.45e−01 (±2.25e−03)(0.00e+00)
f_{14}	**1.05e−01 (±6.07e−04)(−)**	1.31e−01 (±4.84e−04)(0.00e+00)	1.31e−01 (±6.11e−04)(0.00e+00)	1.31e−01 (±4.45e−04)(0.00e+00)
f_{15}	**1.23e+01 (±1.61e−01)(−)**	2.69e+01 (±1.99e−01)(0.00e+00)	2.70e+01 (±2.65e−01)(0.00e+00)	2.68e+01 (±1.98e−01)(0.00e+00)
f_{16}	**3.76e+01 (±4.11e−02)(−)**	3.94e+01 (±2.21e−02)(3.86e−05)	3.94e+01 (±5.01e−02)(7.33e−05)	3.94e+01 (±2.03e−02)(5.34e−05)
f_{17}	**2.23e+03 (±2.01e+01)(−)**	3.87e+03 (±3.81e+01)(0.00e+00)	3.86e+03 (±2.95e+01)(0.00e+00)	3.84e+03 (±3.64e+01)(0.00e+00)
f_{18}	**2.15e+02 (±6.50e−01)(−)**	4.24e+02 (±1.16e+01)(0.00e+00)	4.06e+02 (±2.07e+00)(0.00e+00)	4.10e+02 (±1.39e+01)(0.00e+00)
f_{19}	1.04e+02 (±2.61e−01)(−)	**1.01e+02 (±1.09e+00)(5.23e−03)**	**1.01e+02 (±1.16e+00)(4.16e−03)**	**1.01e+02 (±1.02e+00)(6.43e−03)**
f_{20}	**3.42e+01 (±2.54e−01)(−)**	1.56e+02 (±1.54e+00)(0.00e+00)	1.54e+02 (±2.61e+00)(0.00e+00)	1.53e+02 (±1.45e+00)(0.00e+00)
f_{21}	**6.60e+02 (±9.26e+00)(−)**	3.57e+03 (±5.51e+02)(0.00e+00)	3.30e+03 (±3.46e+01)(0.00e+00)	3.35e+03 (±4.58e+02)(0.00e+00)
f_{22}	**8.88e+02 (±1.22e+01)(−)**	1.27e+03 (±9.70e+00)(9.45e−15)	1.27e+03 (±9.57e+00)(8.10e−15)	1.27e+03 (±8.81e+00)(7.45e−15)
f_{23}	1.23e+02 (±0.00e+00)(−)	**1.22e+02 (±0.00e+00)(2.23e−03)**	**1.22e+02 (±0.00e+00)(2.23e−03)**	**1.22e+02 (±0.00e+00)(2.23e−03)**
f_{24}	**3.71e+02 (±1.02e−01)(−)**	4.04e+02 (±2.41e−01)(8.45e−06)	4.04e+02 (±3.36e−01)(9.44e−06)	4.04e+02 (±2.32e−01)(7.34e−06)
f_{25}	**2.01e+02 (±8.84e−04)(−)**	**2.01e+02 (±2.52e−03)(1.00e+00)**	**2.01e+02 (±1.17e−03)(1.00e+00)**	**2.01e+02 (±2.46e−03)(1.00e+00)**
f_{26}	**1.00e+02 (±1.32e−03)(−)**	**1.00e+02 (±2.24e−03)(1.00e+00)**	**1.00e+02 (±2.99e−03)(1.00e+00)**	**1.00e+02 (±1.96e−03)(1.00e+00)**
f_{27}	**4.00e+02 (±3.75e−05)(−)**	**4.00e+02 (±1.31e−04)(1.00e+00)**	**4.00e+02 (±1.93e−04)(1.00e+00)**	**4.00e+02 (±1.41e−04)(1.00e+00)**
f_{28}	**8.31e+02 (±1.85e+00)(−)**	8.71e+02 (±4.99e+00)(3.23e−04)	8.64e+02 (±6.23e+00)(1.29e−04)	8.65e+02 (±3.18e+00)(1.56e−04)
f_{29}	**7.13e+02 (±1.36e+00)(−)**	9.04e+02 (±8.55e+00)(3.78e−15)	8.96e+02 (±7.81e+00)(1.33e−15)	8.90e+02 (±8.87e+00)(8.56e−14)
f_{30}	**2.15e+03 (±1.26e+01)(−)**	2.88e+03 (±2.12e+01)(6.43e−05)	2.89e+03 (±2.31e+01)(9.66e−05)	2.87e+03 (±2.04e+01)(4.67e−05)

A.3 DDMS−TEDA *vs.* Recent Sequential EAs

We have also compared DDMS−TEDA against some recent sequential EAs, namely L−SHADE−EpSin (Awad et al., 2016), FLDE (Tan et al., 2021), MSHCSA (Zhang et al., 2019a), and ADECSA (Wang et al., 2022).

L−SHADE−EpSin is an algorithm which uses a ensemble sinusoidal approach to automatically adapt the values of the scaling factor of the DE (Awad et al., 2016). The approach consists of a mixture of two sinusoidal formulas: a non-adaptive sinusoidal decreasing adjustment and an adaptive history-based sinusoidal increasing adjustment. The idea is to find an effective balance between the exploitation of the already found best solutions, and the exploration of non-visited regions. The work demonstrated the efficiency and robustness of the L−SHADE−EpSin to obtain better results when compared to L−SHADE algorithm and other state-of-the-art algorithms.

FLDE is an DE-based algorithm which applies an adaptive mutation strategy based on fitness landscape (Tan et al., 2021). First, the algorithm analyzes the fitness landscape features of each benchmark function. Then, the relationship between three mutation strategies and fitness landscape features is trained by random forest (RF) offline. Finally, the trained RF is used to predict which mutation strategy should be utilized to perform mutation operator for each problem during the evolutionary process. The authors showed that FLDE was highly competitive when compared with others DE algorithms.

MSHCSA is an algorithm that proposes a modified combinatorial recombination to bring diversity to the population and avoid premature convergence (Zhang et al., 2019a). A success-history based adaptive mutation strategy is introduced to form a selection algorithm that improves the search ability. The mutation operator is also modified and analyzed through experimental comparison. To cope with the stagnation, the gene knockout strategy is proposed. The experimental results showed that MSHCSA is quite competitive when compared with some state-of-the-art algorithms.

The recent work presented in Wang et al. (2022) propose an improved clonal selection algorithm called an adaptive clonal selection algorithm with multiple differential evolution strategies (ADECSA). The algorithm has three features: (1) an adaptive mutation strategy pool based on its historical records of success is introduced to guide the immune response process effectively; (2) an adaptive population resizing method is adopted to speed up convergence; and (3) a premature convergence detection method and a stagnation detection method are proposed to alleviate premature convergence and stagnation problems in the evolution by enhancing the diversity of the population. The experimental results showed that ADECSA has the best overall performance compared with other clonal algorithms and DE-based algorithms.

The error values ($f(\mathbf{x}) - f(\mathbf{x}^*)$) are given in Tables A.10 and A.11 for 50 and 100 variables, respectively. The numbers in parentheses represent the standard error of each algorithm. The results of the L-SHADE EpSin, FLDE, MSHCSA, and ADECSA are extracted from Tan et al. (2021) and Wang et al. (2022), respectively. We added a bold highlight to the strategies that obtained the lowest average error for each function.

Table A.10 shows that DDMS-TEDA wins on 18 functions when compared with DE-based algorithms (L-SHADE EpSin and FLDE) with 50 variables. DDMS-TEDA is overcome by L-SHADE EpSin or FLDE in 9 occasions. Table A.11 shows the corresponding results with 100 variables. For the average performance, it outperforms other DE-based algorithms in 19 functions. These results demonstrate that DDMS-TEDA has the potential to improve L-SHADE and make it outperforms recent DE-based sequential algorithms.

When compared with recent clonal selection algorithms (MSHCSA and ADECSA), DDMS-TEDA wins on 13 and 11 functions considering 50 and 100 variables, respectively. Regarding the recent ADECSA, which demonstrated excellent performance in (Wang et al., 2022) when compared with DE-based and clonal selection algorithms, DDMS-TEDA loses on 12 and 16 functions and wins in 13 and 11 functions, considering 50 and 100 variables, respectively. Again, these results show the potential of DDMS-TEDA to improve the performance of L-SHADE.

The results demonstrate that the DDMS-TEDA can promote better performance of evolutionary algorithms within a distributed model. In future work, we want to test it using more recent optimizers, such as L-SHADE EpSin, FLDE, and ADECSA, in order to validate this hypothesis.

TABLE A.10: Average error on D = 50: DDMS−TEDA *vs.* some recent sequential EAs. Numbers in bold highlight represent the strategies with the lowest average error. Numbers in parentheses represent the standard error.

Fun.	DDMS−TEDA	L−SHADE EpSin (Awad et al., 2016)	FLDE (Tan et al., 2021)	MSHCSA (Zhang et al., 2019a)	ADECSA (Wang et al., 2022)
f_1	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	1.05e+03 (±1.87e+03)
f_2	**0.00e+00 (±0.00e+00)**	1.57e+00 (±1.93e+00)	**0.00e+00 (±0.00e+00)**	2.51e−07 (±1.77e−07)	3.31e−08 (±9.65e−08)
f_3	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**
f_4	**1.36e+01 (±1.10e+00)**	5.14e+01 (±4.43e+01)	5.77e+01 (±5.05e+01)	2.72e+01 (±8.12e−01)	5.30e+01 (±4.83e+01)
f_5	2.02e+01 (±3.09e−03)	2.52e+01 (±6.44e+00)	**1.14e+01 (±2.61e+00)**	2.00e+01 (±2.44e−04)	2.02e+01 (±1.35e−01)
f_6	1.27e−05 (±9.58e−07)	9.16e−07 (±1.07e−06)	**0.00e+00 (±0.00e+00)**	5.09e+01 (±1.79e+00)	3.46e−02 (±1.29e−01)
f_7	**0.00e+00 (±0.00e+00)**	7.66e+01 (±6.06e+00)	6.32e+01 (±1.71e+00)	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**
f_8	1.04e−08 (±7.86e−10)	2.63e+01 (±6.59e+00)	1.13e+01 (±2.14e+00)	3.26e+01 (±3.80e+00)	**0.00e+00 (±0.00e+00)**
f_9	1.42e+01 (±1.63e−01)	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	7.68e+01 (±8.82e+00)	2.28e+01 (±9.51e+00)
f_{10}	2.85e−01 (±3.09e−03)	3.20e+03 (±3.40e+02)	3.04e+03 (±3.37e+02)	1.51e+03 (±2.24e+02)	**4.77e−02 (±1.65e−01)**
f_{11}	2.68e+03 (±2.62e+01)	**2.14e+01 (±2.09e+00)**	4.92e+01 (±1.07e+01)	5.79e+03 (±4.10e+02)	3.64e+03 (±5.63e+02)
f_{12}	**1.60e−01 (±1.13e−03)**	1.48e+03 (±3.65e+02)	2.33E+03 (±5.97E+02)	6.82e−01 (±1.02e−01)	1.84e−01 (±7.37e−02)
f_{13}	**1.17e−01 (±1.17e−03)**	6.94e+01 (±3.45e+01)	6.86e+01 (±3.24e+01)	2.84e−01 (±2.99e−02)	1.73e−01 (±2.49e−02)
f_{14}	2.59e−01 (±4.76e−04)	2.65e+01 (±2.49e+00)	3.02E+01 (±3.47E+00)	2.22e−01 (±4.02e−02)	**1.81e−01 (±2.54e−02)**
f_{15}	**4.69e+00 (±2.45e−02)**	2.56e+01 (±4.06e+00)	4.07e+01 (±1.01e+01)	9.19e+00 (±1.24e+00)	5.16e+00 (±5.11e−01)
f_{16}	**1.57e+01 (±2.44e−02)**	2.75e+02 (±9.97e+01)	3.92e+02 (±1.20e+02)	2.03e+01 (±3.24e−01)	1.66e+01 (±1.01e+00)
f_{17}	3.48e+02 (±8.77e+00)	2.77e+02 (±7.31e+01)	**2.49e+02 (±7.94e+01)**	2.64e+03 (±2.99e+02)	4.70e+02 (±1.87e+02)
f_{18}	9.00e+01 (±4.33e−01)	2.43e+01 (±2.12e+00)	4.52e+01 (±1.46e+01)	7.78e+01 (±1.02e+01)	**2.17e+01 (±6.91e+00)**
f_{19}	8.57e+00 (±9.36e−02)	1.74e+01 (±2.47e+00)	2.70e+01 (±9.17e+00)	1.96e+01 (±1.56e+00)	**8.45e+00 (±1.40e+00)**
f_{20}	**8.06e+00 (±1.18e−01)**	1.14e+02 (±3.55e+01)	1.63e+02 (±6.98e+01)	4.89e+01 (±5.76e+00)	8.63e+00 (±2.46e+00)
f_{21}	3.57e+02 (±4.74e+00)	2.27e+02 (±7.06e+00)	**2.14e+02 (±2.29e+00)**	1.50e+03 (±2.48e+02)	4.06e+02 (±1.49e+02)
f_{22}	1.36e+02 (±1.84e+00)	1.59e+03 (±1.67e+03)	1.13e+03 (±1.65e+03)	4.63e+02 (±1.32e+02)	**1.09e+02 (±8.99e+01)**
f_{23}	**1.24e+02 (±0.00e+00)**	4.39e+02 (±6.90e+00)	4.31e+02 (±3.74e+00)	3.37e+02 (±2.85e−13)	2.00e+02 (±0.00e+00)
f_{24}	2.70e+02 (±8.38e−02)	5.13e+02 (±5.59e+00)	5.09e+02 (±2.60e+00)	2.64e+02 (±3.30e−01)	**2.00e+02 (±2.45e−13)**
f_{25}	**2.00e+02 (±2.19e−04)**	4.80e+02 (±1.08e+00)	4.83e+02 (±5.90e+00)	**2.00e+02 (±1.59e−02)**	**2.00e+02 (±0.00e+00)**
f_{26}	**1.00e+02 (±6.96e−04)**	1.20e+03 (±1.19e+02)	1.18e+03 (±4.45e+01)	**1.00e+02 (±2.83e−02)**	**1.00e+02 (±2.45e−02)**
f_{27}	4.00e+02 (±3.21e−05)	5.25e+02 (±9.21e+00)	5.24e+02 (±8.85e+00)	1.73e+03 (±3.70e+01)	**2.00e+02 (±8.34e−10)**
f_{28}	5.73e+02 (±4.29e−01)	4.59e+02 (±1.19e+01)	4.59e+02 (±3.81e−02)	3.57e+02 (±4.41e−01)	**2.00e+02 (±0.00e+00)**
f_{29}	7.48e+02 (±3.58e+00)	3.53e+02 (±9.78e+00)	3.48e+02 (±9.74e+00)	2.29e+02 (±8.98e−01)	**2.00e+02 (±0.00e+00)**
f_{30}	6.23e+02 (±1.44e+00)	6.58e+05 (±7.24e+04)	6.14e+05 (±4.38e+04)	9.58e+02 (±1.77e+02)	**2.00e+02 (±0.00e+00)**

TABLE A.11: Average error on D = 100: DDMS−TEDA *vs.* some recent sequential EAs. Numbers in bold highlight represent the strategies with the lowest average error. Numbers in parentheses represent the standard error.

Fun.	DDMS−TEDA	L−SHADE EpSin (Awad et al., 2016)	FLDE (Tan et al., 2021)	MSHCSA (Zhang et al., 2019a)	ADECSA (Wang et al., 2022)
f_1	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	9.95e+04 (±4.26e+04)	1.40e+05 (±4.94e+04)
f_2	**0.00e+00 (±0.00e+00)**	3.57e+11 (±6.18e+11)	2.35e+11 (±5.10e+11)	2.46e+00 (±1.21e+00)	4.60e−03 (±4.99e−03)
f_3	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	6.59e−07 (±3.64e−07)	**0.00e+00 (±0.00e+00)**
f_4	9.34e+01 (±2.50e+00)	1.99e+02 (±8.30e+00)	2.01e+02 (±7.59e+00)	**8.44e+01 (±6.18e−01)**	1.82e+02 (±2.97e+01)
f_5	2.04e+01 (±1.33e−02)	5.59e+01 (±9.91e+00)	3.55e+01 (±5.58e+00)	**2.00e+01 (±1.87e−03)**	2.05e+01 (±1.74e−01)
f_6	7.01e+00 (±6.11e−02)	6.02e−05 (±2.18e−05)	**0.00e+00 (±0.00e+00)**	1.29e+02 (±2.61e+00)	5.07e−01 (±6.34e−01)
f_7	**0.00e+00 (±0.00e+00)**	1.62e+02 (±7.91e+00)	1.41e+02 (±5.48e+00)	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**
f_8	2.86e−01 (±1.33e−02)	5.35e+01 (±5.39e+00)	3.55e+01 (±3.64e+00)	2.36e+02 (±1.64e+01)	**8.78e−06 (±1.78e−05)**
f_9	4.08e+01 (±4.02e−01)	**0.00e+00 (±0.00e+00)**	**0.00e+00 (±0.00e+00)**	3.17e+02 (±1.82e+01)	4.78e+01 (±1.66e+01)
f_{10}	5.52e+01 (±2.94e−01)	1.03e+04 (±5.21e+02)	1.07e+04 (±4.74e+02)	1.11e+04 (±5.54e+02)	**1.13e+00 (±5.21e−01)**
f_{11}	9.62e+03 (±2.30e+01)	**4.92e+01 (±3.02e+02)**	4.34e+02 (±1.31e+02)	1.69e+04 (±5.54e+02)	1.01e+04 (±1.10e+03)
f_{12}	3.33e−01 (±2.06e−03)	4.62e+03 (±6.48e+02)	2.34e+04 (±8.91e+03)	1.33e+00 (±1.34e−01)	**3.31e−01 (±4.74e−02)**
f_{13}	**2.11e−01 (±5.07e−04)**	1.25e+02 (±3.65e+01)	5.61e+02 (±2.73e+02)	3.67e−01 (±3.12e−02)	3.05e−01 (±2.58e−02)
f_{14}	**1.05e−01 (±6.07e−04)**	4.97e+01 (±8.17e+00)	2.45e+02 (±3.34e+01)	3.12e−01 (±9.89e−02)	2.19e−01 (±1.63e−02)
f_{15}	**1.23e+01 (±1.61e−01)**	8.99e+01 (±2.83e+01)	2.37e+02 (±6.18e+01)	3.00e+01 (±1.84e+00)	1.54e+01 (±9.47e−01)
f_{16}	**3.76e+01 (±4.11e−02)**	1.82e+03 (±2.36e+02)	1.53e+03 (±2.50e+02)	4.35e+01 (±4.63e−01)	3.84e+01 (±6.95e−01)
f_{17}	2.23e+03 (±2.01e+01)	1.32e+03 (±1.74e+02)	**1.24e+03 (±1.26e+02)**	5.87e+03 (±6.98e+02)	2.99e+03 (±5.52e+02)
f_{18}	2.15e+02 (±6.50e−01)	**7.79e+01 (±1.99e+01)**	1.97e+02 (±3.95e+01)	2.82e+02 (±9.44e+01)	1.46e+02 (±2.39e+01)
f_{19}	1.04e+02 (±2.61e−01)	9.55e+01 (±6.05e+00)	1.68e+02 (±1.52e+01)	**4.31e+01 (±3.35e−01)**	8.89e+01 (±1.33e+00)
f_{20}	3.42e+01 (±2.54e−01)	1.08e+03 (±2.16e+02)	1.59e+03 (±1.70e+02)	2.18e+02 (±1.49e+01)	**3.16e+01 (±5.79e+00)**
f_{21}	6.60e+02 (±9.26e+00)	2.77e+02 (±6.94e+00)	**2.60e+02 (±4.13e+00)**	4.76e+03 (±3.41e+02)	6.30e+02 (±2.55e+02)
f_{22}	**8.88e+02 (±1.22e+01)**	1.04e+04 (±5.30e+02)	1.14e+04 (±6.65e+02)	1.53e+03 (±3.35e+02)	9.52e+02 (±2.50e+02)
f_{23}	**1.23e+02 (±0.00e+00)**	5.98e+02 (±7.69e+00)	5.68e+02 (±7.79e+00)	3.45e+02 (±1.05e−12)	2.00e+02 (±0.00e+00)
f_{24}	3.71e+02 (±1.02e−01)	9.17e+02 (±1.34e+01)	9.14e+02 (±8.57e+00)	3.84e+02 (±2.91e+00)	**2.00e+02 (±2.45e−13)**
f_{25}	2.01e+02 (±8.84e−04)	6.84e+02 (±4.34e+01)	7.34e+02 (±4.18e+01)	2.01e+02 (±1.70e−01)	**2.00e+02 (±0.00e+00)**
f_{26}	**1.00e+02 (±1.32e−03)**	3.11e+03 (±1.22e+02)	3.34e+03 (±1.07e+02)	**1.00e+02 (±2.89e−02)**	**1.00e+02 (±2.45e−02)**
f_{27}	4.00e+02 (±3.75e−05)	5.89e+02 (±1.31e+01)	6.40e+02 (±2.17e+01)	3.71e+03 (±5.61e+01)	**2.00e+02 (±7.83e−10)**
f_{28}	8.31e+02 (±1.85e+00)	5.25e+02 (±2.20e+01)	5.24e+02 (±1.36e+01)	4.04e+02 (±5.30e+00)	**2.00e+02 (±0.00e+00)**
f_{29}	7.13e+02 (±1.36e+00)	1.12e+03 (±1.49e+02)	1.32e+03 (±1.95e+02)	2.43e+02 (±4.22e+00)	**2.00e+02 (±0.00e+00)**
f_{30}	2.15e+03 (±1.26e+01)	2.46e+03 (±1.44e+02)	2.41e+03 (±1.78e+02)	2.85e+03 (±2.73e+02)	**2.00e+02 (±0.00e+00)**

Bibliography

Abdelhafez, A., Alba, E., and Luque, G. (2019). Performance analysis of synchronous and asynchronous distributed genetic algorithms on multiprocessors. *Swarm and Evolutionary Computation*, 49:147–157.

Ali, M. Z., Awad, N. H., and Suganthan, P. N. (2015). Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization. *Applied Soft Computing*, 33:304–327.

Ali, M. Z., Awad, N. H., Suganthan, P. N., and Reynolds, R. G. (2016). An adaptive multi-population differential evolution with dynamic population reduction. *IEEE transactions on cybernetics*, 47(9):2768–2779.

Angelov, P. (2014). Outside the box: an alternative data analytics framework. *Journal of Automation Mobile Robotics and Intelligent Systems*, 8(2):29–35.

Apolloni, J., García-Nieto, J., Alba, E., and Leguizamón, G. (2014). Empirical evaluation of distributed differential evolution on standard benchmarks. *Applied Mathematics and Computation*, 236:351–366.

Apolloni, J., Leguizamón, G., García-Nieto, J., and Alba, E. (2008). Island based distributed differential evolution: an experimental study on hybrid testbeds. In *2008 Eighth International Conference on Hybrid Intelligent Systems*, pages 696–701. IEEE.

Arabas, J. and Opara, K. (2019). Population diversity of nonelitist evolutionary algorithms in the exploration phase. *IEEE Transactions on Evolutionary Computation*, 24(6):1050–1062.

Araujo, L. and Merelo, J. J. (2010). Diversity through multiculturality: Assessing migrant choice policies in an island model. *IEEE Transactions on Evolutionary Computation*, 15(4):456–469.

Awad, N. H., Ali, M. Z., and Suganthan, P. N. (2018). Ensemble of parameters in a sinusoidal differential evolution with niching-based population reduction. *Swarm and evolutionary computation*, 39:141–156.

Awad, N. H., Ali, M. Z., Suganthan, P. N., and Reynolds, R. G. (2016). An ensemble sinusoidal parameter adaptation incorporated with l-shade for solving cec2014 benchmark problems. In *2016 IEEE congress on evolutionary computation (CEC)*, pages 2958–2965. IEEE.

Bezerra, C. G., Costa, B. S. J., Guedes, L. A., and Angelov, P. P. (2016). A new evolving clustering algorithm for online data streams. In *2016 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 162–168. IEEE.

Cabrera, D. M. (2016). Evolutionary algorithms for large-scale global optimisation: a snapshot, trends and challenges. *Progress in Artificial Intelligence*, 5(2):85–89.

Chen, C., Yan, Y., and Liu, Q. (2022). An adaptive differential evolution with extended historical memory and iterative local search. *Applied Soft Computing*, 125:109203.

Chen, S., Montgomery, J., and Bolufé-Röhler, A. (2015). Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Applied Intelligence*, 42(3):514–526.

Črepinšek, M., Liu, S.-H., and Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 45(3):35.

De Falco, I., Scafuri, U., Tarantino, E., and Della Cioppa, A. (2007). A distributed differential evolution approach for mapping in a grid environment. In *15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing (PDP'07)*, pages 442–449. IEEE.

Dinno, A. (2015). Nonparametric pairwise multiple comparisons in independent groups using dunn's test. *The Stata Journal*, 15(1):292–300.

Drugan, M. M. (2019). Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. *Swarm and evolutionary computation*, 44:228–246.

Duarte, G. R., de Castro Lemonge, A. C., da Fonseca, L. G., and de Lima, B. S. L. P. (2021). An island model based on stigmergy to solve optimization problems. *Natural Computing*, 20(3):413–441.

Eiben, A. E. and Smith, J. E. (2015). *Introduction to Evolutionary Computing*. Springer Publishing Company, Incorporated, 2nd edition.

Ge, Y.-F., Yu, W.-J., Lin, Y., Gong, Y.-J., Zhan, Z.-H., Chen, W.-N., and Zhang, J. (2017). Distributed differential evolution based on adaptive mergence and split for large-scale optimization. *IEEE transactions on cybernetics*, 48(7):2166–2180.

Gong, Y.-J., Chen, W.-N., Zhan, Z.-H., Zhang, J., Li, Y., Zhang, Q., and Li, J.-J. (2015). Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286–300.

Halim, A. H., Ismail, I., and Das, S. (2021). Performance assessment of the metaheuristic optimization algorithms: an exhaustive review. *Artificial Intelligence Review*, 54:2323–2409.

Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., and Prasath, V. S. (2019). Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, 10(12):390.

Hu, X.-M., He, F.-L., Chen, W.-N., and Zhang, J. (2017). Cooperation coevolution with fast interdependency identification for large scale optimization. *Information Sciences*, 381:142–160.

Ishimizu, T. and Tagawa, K. (2010). A structured differential evolution for various network topologies. *International Journal of Computers and Communications*, 4(1):2–8.

Jia, Y.-H., Chen, W.-N., Gu, T., Zhang, H., Yuan, H.-Q., Kwong, S., and Zhang, J. (2018). Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):188–202.

Jian, J.-R., Zhan, Z.-H., and Zhang, J. (2020). Large-scale evolutionary optimization: a survey and experimental comparative study. *International Journal of Machine Learning and Cybernetics*, 11(3):729–745.

Karafotias, G., Hoogendoorn, M., and Eiben, Á. E. (2014). Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187.

Karniadakis, G., Karniadakis, G. E., and Kirby II, R. M. (2003). *Parallel scientific computing in C++ and MPI: a seamless approach to parallel algorithms and their implementation*, volume 1. Cambridge University Press.

Kazimipour, B., Li, X., and Qin, A. K. (2013). Initialization methods for large scale global optimization. In *2013 IEEE Congress on Evolutionary Computation*, pages 2750–2757. IEEE.

LaTorre, A., Muelas, S., and Peña, J.-M. (2015). A comprehensive comparison of large scale global optimizers. *Information Sciences*, 316:517–549.

Li, J. and Gonsalves, T. (2022). Parallel hybrid island metaheuristic algorithm. *IEEE Access*, 10:42268–42286.

Li, J.-Y., Du, K.-J., Zhan, Z.-H., Wang, H., and Zhang, J. (2022). Distributed differential evolution with adaptive resource allocation. *IEEE transactions on cybernetics*.

Li, L., Fang, W., Wang, Q., and Sun, J. (2019). Differential grouping with spectral clustering for large scale global optimization. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 334–341. IEEE.

Li, X., Tang, K., Omidvar, M. N., Yang, Z., Qin, K., and China, H. (2013). Benchmark functions for the cec 2013 special session and competition on large-scale global optimization. *gene*, 7(33):8.

Liang, J. J., Qu, B. Y., and Suganthan, P. N. (2013). Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 635.

Lopes, R. A. and de Freitas, A. R. (2017). Island-cellular model differential evolution for large-scale global optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1841–1848.

Lopes, R. A., Silva, R. C., and de Freitas, A. R. (2021). An abstract interface for large-scale continuous optimization decomposition methods. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1267–1274.

Lorion, Y., Bogon, T., Timm, I. J., and Drobnik, O. (2009). An agent based parallel particle swarm optimization-appso. In *2009 IEEE Swarm Intelligence Symposium*, pages 52–59. IEEE.

Lynn, N., Ali, M. Z., and Suganthan, P. N. (2018). Population topologies for particle swarm optimization and differential evolution. *Swarm and evolutionary computation*, 39:24–35.

Mahdavi, S., Shiri, M. E., and Rahnamayan, S. (2015). Metaheuristics in large-scale global continues optimization: A survey. *Information Sciences*, 295:407–428.

Maia, J., Junior, C. A. S., Guimarães, F. G., de Castro, C. L., Lemos, A. P., Galindo, J. C. F., and Cohen, M. W. (2020). Evolving clustering algorithm based on mixture of typicalities for stream data mining. *Future Generation Computer Systems*, 106:672–684.

McGinley, B., Maher, J., O'Riordan, C., and Morgan, F. (2011). Maintaining healthy population diversity using adaptive crossover, mutation, and selection. *IEEE Transactions on Evolutionary Computation*, 15(5):692–714.

Mei, Y., Omidvar, M. N., Li, X., and Yao, X. (2016). A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Transactions on Mathematical Software (TOMS)*, 42(2):1–24.

Meng, Q., Wu, J., Ellis, J., and Kennedy, P. J. (2017). Dynamic island model based on spectral clustering in genetic algorithm. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1724–1731. IEEE.

Molina, D., LaTorre, A., and Herrera, F. (2018). Shade with iterative local search for large-scale global optimization. In *2018 IEEE congress on evolutionary computation (CEC)*, pages 1–8. IEEE.

Omidvar, M. N., Li, X., Mei, Y., and Yao, X. (2013). Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on evolutionary computation*, 18(3):378–393.

Omidvar, M. N., Li, X., and Yao, X. (2021a). A review of population-based metaheuristics for large-scale black-box global optimization—part i. *IEEE Transactions on Evolutionary Computation*, 26(5):802–822.

Omidvar, M. N., Li, X., and Yao, X. (2021b). A review of population-based metaheuristics for large-scale black-box global optimization—part ii. *IEEE Transactions on Evolutionary Computation*, 26(5):823–843.

Omidvar, M. N., Yang, M., Mei, Y., Li, X., and Yao, X. (2017). Dg2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE Transactions on Evolutionary Computation*, 21(6):929–942.

Peng, F., Tang, K., Chen, G., and Yao, X. (2009). Multi-start jade with knowledge transfer for numerical optimization. In *2009 IEEE Congress on Evolutionary Computation*, pages 1889–1895. IEEE.

Piotrowski, A. P. and Napiorkowski, J. J. (2018). Step-by-step improvement of jade and shade-based algorithms: Success or failure? *Swarm and evolutionary computation*, 43:88–108.

Piotrowski, A. P., Napiorkowski, J. J., and Kiczko, A. (2012). Differential evolution algorithm with separated groups for multi-dimensional optimization problems. *European Journal of Operational Research*, 216(1):33–46.

Price, D. and Radaideh, M. I. (2023). Animorphic ensemble optimization: a large-scale island model. *Neural Computing and Applications*, 35(4):3221–3243.

Saw, J. G., Yang, M. C., and Mo, T. C. (1984). Chebyshev inequality with estimated mean and variance. *The American Statistician*, 38(2):130–132.

Smit, S. K. and Eiben, A. E. (2009). Comparing parameter tuning methods for evolutionary algorithms. In *2009 IEEE congress on evolutionary computation*, pages 399–406. IEEE.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Sudholt, D. (2015). Parallel evolutionary algorithms. In *Springer Handbook of Computational Intelligence*, pages 929–959. Springer.

Sudholt, D. (2020). The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses. In *Theory of Evolutionary Computation*, pages 359–404. Springer.

Sun, Y., Kirley, M., and Halgamuge, S. K. (2017). A recursive decomposition method for large scale continuous optimization. *IEEE Transactions on Evolutionary Computation*, 22(5):647–661.

Sun, Y., Li, X., Ernst, A., and Omidvar, M. N. (2019). Decomposition for large-scale optimization problems with overlapping components. In *2019 IEEE congress on evolutionary computation (CEC)*, pages 326–333. IEEE.

Sun, Y., Omidvar, M. N., Kirley, M., and Li, X. (2018). Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition. In *Proceedings of the genetic and evolutionary computation conference*, pages 889–896.

Tan, Z., Li, K., and Wang, Y. (2021). Differential evolution with adaptive mutation strategy based on fitness landscape analysis. *Information Sciences*, 549:142–163.

Tanabe, R. and Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. In *2013 IEEE congress on evolutionary computation*, pages 71–78. IEEE.

Tanabe, R. and Fukunaga, A. S. (2014). Improving the search performance of shade using linear population size reduction. In *2014 IEEE congress on evolutionary computation (CEC)*, pages 1658–1665. IEEE.

Thukral, A. K., Bhardwaj, R., Kumar, V., and Sharma, A. (2019). New indices regarding the dominance and diversity of communities, derived from sample variance and standard deviation. *Heliyon*, 5(10):e02606.

Wang, T.-C., Lin, C.-Y., Liaw, R.-T., and Ting, C.-K. (2019). Empirical analysis of island model on large scale global optimization. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 342–349. IEEE.

Wang, Y., Li, T., Liu, X., and Yao, J. (2022). An adaptive clonal selection algorithm with multiple differential evolution strategies. *Information Sciences*, 604:142–169.

Weber, M., Neri, F., and Tirronen, V. (2009). Distributed differential evolution with explorative–exploitative population families. *Genetic Programming and Evolvable Machines*, 10(4):343.

Yang, M., Li, C., Cai, Z., and Guan, J. (2014). Differential evolution with auto-enhanced population diversity. *IEEE transactions on cybernetics*, 45(2):302–315.

Yang, M., Omidvar, M. N., Li, C., Li, X., Cai, Z., Kazimipour, B., and Yao, X. (2016). Efficient resource allocation in cooperative co-evolution for large-scale global optimization. *IEEE Transactions on Evolutionary Computation*, 21(4):493–505.

Yang, Z., Tang, K., and Yao, X. (2008). Self-adaptive differential evolution with neighborhood search. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1110–1116. IEEE.

Zhan, Z.-H., Shi, L., Tan, K. C., and Zhang, J. (2022). A survey on evolutionary computation for complex continuous optimization. *Artificial Intelligence Review*, 55(1):59–110.

Zhang, C., Chen, J., and Xin, B. (2013). Distributed memetic differential evolution with the synergy of lamarckian and baldwinian learning. *Applied Soft Computing*, 13(5):2947–2959.

Zhang, J. and Sanderson, A. C. (2009a). Adaptive differential evolution: A robust approach to multimodal problem optimization. Berlin Springer.

Zhang, J. and Sanderson, A. C. (2009b). Jade: adaptive differential evolution with optional external archive. *IEEE Transactions on evolutionary computation*, 13(5):945–958.

Zhang, W., Gao, K., Zhang, W., Wang, X., Zhang, Q., and Wang, H. (2019a). A hybrid clonal selection algorithm with modified combinatorial recombination and success-history based adaptive mutation for numerical optimization. *Applied Intelligence*, 49:819–836.

Zhang, X.-Y., Gong, Y.-J., Lin, Y., Zhang, J., Kwong, S., and Zhang, J. (2019b). Dynamic cooperative coevolution for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 23(6):935–948.