Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Dynamically Reconfigurable Heterogeneous Parallel Island Model

(Modelo de Ilhas Paralelo Heterogêneo Dinamicamente Reconfigurável)

Lucas Ângelo da Silveira

Tese apresentada como requisito parcial para conclusão do Doutorado em Informática

Orientador
Prof. Dr. Mauricio Ayala-Rincón

Brasília
2024

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Dynamically Reconfigurable Heterogeneous Parallel Island Model
(Modelo de Ilhas Paralelo Heterogêneo Dinamicamente Reconfigurável)

Lucas Ângelo da Silveira

Tese apresentada como requisito parcial para
conclusão do Doutorado em Informática

Brasília, 26 de abril de 2024

Dedicatória

Dedico a minha família.

Agradecimentos

Gostaria de expressar minha gratidão ao meu orientador, Mauricio Ayala-Rincón, cuja orientação e apoio foram fundamentais para a realização deste trabalho. À Thaynara Adrielly de Lima, minha parceira de pesquisa, agradeço pela colaboração e contribuições valiosas que enriqueceram este projeto.

Agradeço também à Fundação de Apoio à Pesquisa do Distrito Federal (FAPDF) pelo financiamento das viagens para a apresentação de artigos, possibilitando a divulgação e o aperfeiçoamento desta pesquisa.

Em especial, gostaria de expressar minha gratidão aos meus pais e à minha esposa, verdadeiros alicerces ao longo de toda esta jornada acadêmica. Suas palavras de incentivo, amor incondicional e apoio constante foram a força motriz por trás de cada passo dado nesta trajetória desafiadora.

À Universidade de Brasília, estendo meu agradecimento pela oportunidade de aprendizado proporcionada, pelos recursos disponíveis e pelo ambiente acadêmico estimulante, que contribuíram significativamente para o amadurecimento deste trabalho.

O êxito alcançado nesta pesquisa é resultado direto da sabedoria e orientação dos notáveis indivíduos que generosamente compartilharam seus conhecimentos, abrindo horizontes além da minha visão inicial. Expresso meu mais sincero reconhecimento a todos que, de diversas formas, contribuíram para o desenvolvimento deste projeto. Seu apoio e colaboração foram fundamentais para o alcance dos resultados apresentados, e por isso, sou profundamente grato.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Problemas de otimização $\mathcal{NP}$-difíceis são encontrados em diversos campos de atividade, e à medida que a compreensão e a prática nesses campos avançam, suas complexidades se acentuam. Nas últimas décadas, diversos algoritmos bioinspirados para resolver problemas de otimização foram propostos. Cada um desses algoritmos possui características únicas que impactam de maneiras distintas tanto o processo evolutivo quanto a qualidade das soluções alcançadas. O Modelo de Ilhas Paralelas (MIP) é uma estratégia de paralelização de algoritmos bioinspirados que proporciona ganhos significativos na acurácia das soluções. Nesse modelo, o conjunto de soluções candidatas é dividido em subpopulações denominadas ilhas. Cada ilha evolui seu conjunto de soluções por meio de seu próprio algoritmo bioinspirado, operando de forma paralela às outras ilhas. Periodicamente, as ilhas trocam soluções entre si através do processo de migração. Esse movimento de soluções entre as ilhas é condicionado à topologia do modelo e a um conjunto de regras que compõem a política de migração. Este trabalho propõe uma nova abordagem de implementação para MIPs, inspirada em heterogeneidade e reconfiguração algorítmica: o MIP heterogêneo com reconfiguração baseada em estagnação. A heterogeneidade permite a execução de diferentes algoritmos bioinspirados nas ilhas, aumentando a diversidade. Ao mesmo tempo, a reconfiguração algorítmica permite a substituição dos algoritmos bioinspirados caso a estagnação das ilhas é detectada. Durante o processo evolutivo, cada ilha mantém um registro do seu progresso, mensurado pelo desempenho do melhor indivíduo em cada ilha, na geração atual e nas duas gerações anteriores. Sempre que exista *estagnação*, ou seja, não se detecte progresso, a ilha é *reconfigurada* para continuar o processo evolutivo executando o melhor algoritmo bioinspirado até o momento. Essa abordagem é particularmente útil para lidar com problemas de otimização nos quais encontrar soluções ótimas em tempo polinomial é impraticável. O novo modelo com reconfiguração baseada em estagnação computa soluções melhores que as obtidas por uma variedade de MIPs homogêneos e MIPs heterogêneos com reconfiguração aplicada numa frequência fixa.

Palavras-chave: Algoritmos Bioinspirados, Modelos Paralelos de Ilhas, Reconfiguração Algorítmica, Problemas $\mathcal{NP}$-difíceis

Abstract

$\mathcal{NP}$-hard optimization problems are encountered in various fields of activity, and as the understanding and practice in these fields advance, their complexities become more pronounced. Several bioinspired algorithms have been proposed in recent decades to address optimization problems. Each of these algorithms possesses unique characteristics that impact the evolutionary process and the quality of the solutions achieved in distinct ways. The Parallel Island Model (PIM) is a strategy for parallelizing bioinspired algorithms that yields significant gains in solution accuracy. In this model, the set of candidate solutions is divided into subpopulations known as islands. Islands parallelly evolve, and each island modifies its solutions using its bioinspired algorithm. Periodically, islands exchange solutions through the migration process. This movement of solutions between islands is conditioned by the model's topology and a set of rules comprising the migration policy. This work aims to design a model that profits from the PIM architecture to compute the highest quality solutions. A new model is introduced, inspired by heterogeneity and algorithmic reconfiguration: the stagnation-based reconfigurable heterogeneous PIMs. Heterogeneity allows the execution of different bioinspired algorithms on islands, increasing model diversity. At the same time, algorithmic reconfiguration replaces the applied bioinspired algorithm when islands' stagnation is detected. During the evolutionary process, each island maintains a record of its progress, measured by the fitness of the best individual in each island in the current and previous two generations. Whenever an island presents *stagnation*, i.e., no progress is detected, the island is *reconfigured* to continue the evolutionary process by executing the best-bioinspired algorithm up to that point. The new stagnation-based reconfigurable model provides better solutions than various homogeneous PIMs and heterogeneous PIMs with fixed-frequency-based reconfiguration.

Keywords: Bioinspired Algorithms, Parallel Island Models, Algorithmic Reconfiguration, $\mathcal{NP}$-Hard Problems

Sumário

Lista de Figuras

Lista de Tabelas

Chapter 1

Introduction

Bioinspired Algorithms (BAs) represent methods grounded in natural phenomena and principles derived from biological processes, the behavior of organisms, and complex natural systems. These algorithms foster innovative and practical approaches for a wide range of applications. Their remarkable ability to tackle complex challenges and explore unconventional solutions positions them as relevant and valuable tools, driving advancements across various domains and contributing significantly to the technological innovation landscape. Among the most popular in the literature, we can mention Genetic Algorithm [1] based on the principles of natural selection and biological evolution, Particle Swarm Optimization [2] inspired by the bird flocking, fish schooling, and other agents (organisms) that live and move synchronously in groups, Differential Evolution [3] inspired by biological evolution and the difference between solution vectors, and year after year, new algorithms are emerging such as Artificial Bee Colony [4], Social Spider Algorithm [5], Cuckoo Search algorithm [6], the Grey Wolf Optimizer [7], Whale Optimization Algorithm [8]. These are just a few examples of BAs. Each is based on natural principles, such as evolution, swarm behavior, and social interactions between organisms, among others.

Many complex optimization problems in engineering, logistics, and biology cannot be solved by exact optimization methods in polynomial time. For this class of problems, the so-called $\mathcal{NP}$-hard problems, the BAs are often used to find suboptimal solutions.

In addition, there are Parallel Island Models (PIMs), which have received considerable attention since they can use BAs to enhance exploration and exploitation and improve speedup and accuracy (e.g., [9], [10], [11]). The PIMs comprise a set of islands, each containing instances of BAs, developing their populations simultaneously. Periodically, candidate solutions are exchanged between the islands through a process called migration. This transfer of solutions is subject to a set of rules known as migration policy and involves parameters that require careful calibration to achieve satisfactory results. The communication topologies play a significant role in migration parameters and can be classified as static or dynamic. The static topologies establish predetermined connections between the islands, forming fixed neighborhoods during migration. On the other hand, dynamic topologies allow reconfiguring connections throughout the evolutionary cycle, enabling neighborhood variations during migration.

The PIMs approach expands the resolution of $\mathcal{NP}$-hard problems, offering an innovative way to tackle the complexity of these challenging tasks. By emulating the principles of natural evolution, these models provide an effective alternative to seeking high-quality solutions, fostering diversified exploration of the search space. With their adaptability and ability to find feasible suboptimal solutions, the PIMs play a pivotal role in computer science research, paving the way to enhance the quality of solutions for complex problems with a remarkable gain in runtime. This document is dedicated to extensively exploring the application and benefits of PIMs as a central object of study.

1.1 Motivation

Solving $\mathcal{NP}$-hard problems requires algorithmic techniques that do not allow the computation of exact solutions in polynomial time and have been the subject of studies in computer science since before the 1970s when the notions of computational complexity were consolidated. Thus, we have that the solutions we can find for such problems in polynomial time are approximate. The classic problems, Knapsack, Independent Set, Graph

Coloring, Boolean Satisfiability, and Vehicle Routing, are well-studied in the literature and have applications in various fields, including logistics, features selection, operations research, and computer science. They are used as benchmark problems to test and compare the efficiency of different algorithms and optimization techniques. Due to BAs' simple adaptive nature, authors have resorted to various algorithms to compute high-quality solutions for such complex problems.

The evolution of hardware technology has increasingly enabled the use of parallel algorithms. BAs have attracted significant interest from researchers who have been exploring a variety of parallel algorithms seeking to improve the quality of (sequential) solutions. The parallel modeling simplicity of this class of algorithms has allowed a more efficient approach to exploring solutions to complex problems. In this context, a simple and effective way of working with BAs is through PIMs, which make it possible to explore multiple solutions in parallel and BA collaboration, increasing accuracy and speedup.

Two fundamental approaches emerge in PIMs: homogeneous and heterogeneous PIMs, for short, HoPIMs and HePIMs, respectively. Inspired by the analogy of isolated islands within an archipelago, these approaches share the idea that each island represents a set of potential solutions to the given problem. However, they significantly differ in their characteristics and methods. In HePIMs, the islands possess distinct characteristics, potentially incorporating diverse search techniques and BAs. This approach values variety, enriching the search space exploration and enabling different strategies to collaborate in finding high-quality solutions. On the other hand, HoPIMs aim for uniformity across all islands, employing only a BA and identical strategies to explore the search space. These properties promote cohesive exploration and practical information exchange among the islands.

Migration between the islands is a critical component for the success of both HoPIMs and HePIMs. Migration parameters determine the frequency and which solutions are chosen to migrate. Skillful manipulation of these parameters is essential to achieving quality solutions. Therefore, careful selection and adjustment of migration parameters play a crucial role in the quest for high-quality solutions in PIMs, whether through HoPIMs or HePIMs approaches. These parameters balance global exploration of the search space and detailed local exploration, shaping the effectiveness and overall performance of the optimization strategy. Additionally, effectiveness is influenced by carefully selecting the bioinspired algorithm used to address the specific problem and the strategy employed to evolve the global population.

1.2 Contribution

This work's contribution encompasses a thorough analysis of Parallel Island Models, carefully considering the intricacies of the mechanisms that must be managed to extract improved solutions from this methodology.

The research on PIMs began with implementing a Homogeneous PIM (HoPIM) with migration, and its counterpart without migration in [12]. In the latter model, each island evolves independently, and in the end, the fittest individual among all islands is considered the solution. However, the results revealed low-quality solutions in the HoPIM with migration, inferior to the sequential version, prompting an investigation into this performance. This early research phase preceded the experiments conducted in [13], where parameter calibration was introduced to significantly improve the quality of solutions delivered by HoPIMs, surpassing the accuracy provided by the sequential version. It is worth noting that although a parameter calibration approach was adopted, the methodology employed was not complex. A greedy method was chosen to determine parameter values that could deliver improved solutions. This strategic choice emphasizes the importance of adjusting evolutionary and migratory parameters, filling a gap in the literature that often overlooked the significance of these values in the pursuit of quality solutions. Establishing the necessity of such a proactive parameter calibration approach contributes substantially to advancing research in PIMs. Indeed, as evidenced in [14, 15, 16], parameter calibration is essential in developing Heterogeneous PIMs (HePIMs), which improve solutions.

Island communication topologies are crucial in determining how migration can occur, either static or dynamic, and directly impact the solution quality by indicating which islands are involved in the migration process. In [17], a dynamic topology was introduced, in which migration can be adjusted dynamically based on the quality of solutions and the diversity found among the islands during evolution. The topology establishes the communication structure between the islands according to their evolutionary behavior, influencing the effectiveness of the migration process and, consequently, affecting the overall quality of the solutions achieved. The proposal of a dynamic topology represents an advancement compared to conventional static topologies, as discussed in [18, 19, 15, 20, 16].

The initial populations in the islands, as evidenced in [13, 17], were a crucial aspect of the research. The experiments demonstrate that using initial populations from the sequential version leads to more robust and efficient solutions in HoPIMs. This finding significantly enhanced the quality of obtained solutions and has been adopted in all subsequent works.

The comparative analysis between synchronous and asynchronous HoPIMs, discussed in [16], unveiled a critical dependency on the specific migration parameters for each problem. The sensitivity of these parameters was significantly highlighted, emphasizing their direct influence on the efficiency of HoPIMs. This sensitivity underscores the crucial importance of calibrating these parameters, as noted in [12], as experiments showed that the same model, whether synchronous or asynchronous, exhibits distinct behaviors in accuracy and speedup. These observations extend to heterogeneous models as discussed in [14].

Exploring various evolutionary algorithms in [19] provided valuable insights for implementing HePIMs capable of delivering competitive results. Contrary to the prevailing view in the literature, scenarios were identified where HoPIMs outperform HePIMs in accuracy [14]. To improve HePIMs accuracy, we focused on developing island algorithm reconfiguration. Such a reconfiguration methodology allows islands to execute different algorithms during evolution [15]. This reconfiguration approach offers greater dynamism and flexibility, where, during a fixed interval of evolution, the island with the poorest performance undergoes a process in which the evolutionary algorithm is replaced by the algorithm of the island that has shown the best evolution in its population. In a subsequent research stage, the reconfiguration methodology was refined by adopting stagnation-based island reconfiguration [16]. The concept of stagnation is linked to improving the fitness of the fittest individual. This reconfiguration strategy became a highly flexible dynamic process, granting each island autonomy to trigger the reconfiguration process whenever its population stagnates. Unlike the initial proposal of a single reconfiguration at fixed intervals in [15], stagnation-based reconfiguration opens the door to multiple reconfiguration processes. This approach resulted in improved solutions compared to the approach proposed in [15] and demonstrated superiority over all previously proposed versions of HoPIMs and HePIMs.

The approaches used to analyze challenging optimization problems consist of applying hyper-heuristics. Such methods are based on the selection or generation, in a search space of heuristics, of the most adequate (meta)-heuristics to solve optimization problems instead of only searching in the problem's solution space. In a recent survey, Dokeroglu et al. [21] examine the most cited works on hyper-heuristics published in the last twenty years in relevant journals and conferences. The authors classify hyper-heuristics into, not necessarily disjoint, four classes: selection hyper-heuristics, low-level heuristics, target optimization problems, and parallel hyper-heuristics. Considering such a classification, our work can be seen as a low-level heuristic, as it combines bio-inspired algorithms and exact algorithms within island models to compute the fitness of individuals for solving optimization problems. Furthermore, it can be viewed as a parallel hyper-heuristic, restricted to algorithmic-level parallelism, where different (meta)-heuristics are run independently or cooperatively in parallel.

Additionally, the generic taxonomy of multi-population nature-inspired optimization includes the structure of PIMs (e.g., Ha et al. [22]). Essentially, the generic taxonomy of multi-population refers to optimization approaches that divide the population into small sub-populations performing evolutionary operations, such as selection, crossover, and mutation to evolve individuals, enabling the interaction of these sub-populations through merging, communication, and re-division processes. The goals, as expected, are to avoid premature convergence and maintain population diversity to ensure high-quality solutions. The experience obtained in the design and application of multi-population algorithms, usually described using terms such as “parallel”, “cooperative”, “co-evolution”, and “islands”, among others [22], is relevant for the vital purposes in our study. Critical questions presented in multi-population algorithms are related to several issues that arose in the development of our research. For instance, addressing the question how to determine the number of sub-populations allowed us to design PIMs with a balanced number of islands and island population sizes; exploring how to manage communication between sub-populations enabled us to develop practical communication topologies and migration policies; finally, thoroughly researching how to define the search strategy of each sub-population led us to investigate heterogeneous PIMs and dynamic reconfiguration, which, to the best of our knowledge, has not been addressed in previous work. Additionally, addressing another question that arises in multi-population algorithms, how to determine the search area of each sub-population, is of great interest for future research. Tackling this question is crucial for determining an efficient distribution of the search space exploration by island populations, reducing individual redundancy, and promoting a comprehensive exploration of the search space, thereby avoiding convergence to local optima.

In addition to the two taxonomies mentioned above (hyper-heuristics and multi-population models), Chapter 5 on related work presents more precise references comparing our work.

1.3 Work structure

The document is structured in the chapters described below.

Chapter 2 [Theoretical Referential] presents the background regarding BAs, PIMs, and benchmark problems.

Chapter 3 [Overview of Preliminary Research] describes the work conducted during the research, culminating in developing a novel methodology for island models based on heterogeneity and reconfiguration.

Chapter 4 [Stagnation-Based Reconfigurable Heterogeneous Parallel Island Models] discusses the stagnation-based reconfiguration of HePIMs.

Chapter 5 [Related work] reviews related work.

Chapter 6 [Conclusion and Future Work] concludes and discusses possible future work.

The remaining sections of this chapter include a free translation of the introduction into Portuguese.

1.4 Resumo Estendido (Introdução)

A seguir, inclui-se a tradução das seções precedentes a modo de resumo estendido.

Os Algoritmos Bioinspirados (ABs) representam métodos fundamentados em fenômenos e princípios observados na natureza, derivados de processos biológicos, comportamento de organismos e sistemas naturais complexos. Esses algoritmos promovem abordagens inovadoras e eficazes para uma ampla variedade de aplicações. Sua notável capacidade de enfrentar desafios complexos e explorar soluções não convencionais os posiciona como ferramentas relevantes e valiosas, proporcionando avanços em diversas áreas e contribuindo de maneira significativa para o panorama da inovação tecnológica. Entre os mais populares na literatura, podemos citar o Algoritmo Genético [1] baseado nos princípios de seleção natural e evolução biológica, Otimização por Enxame de Partículas [2] inspirado no comportamento de bandos de pássaros, cardumes de peixes e outros agentes (organismos) que vivem e se movem sincronicamente em grupos, Evolução Diferencial [3] inspirado na evolução biológica e na diferença entre vetores de solução, e ano após ano, novos algoritmos estão surgindo, como Colônia Artificial de Abelhas [4], Algoritmo da Aranha Social [5], Algoritmo de Busca do Cucu [6], Otimizador do Lobo Cinzento [7], Algoritmo de Otimização da Baleia [8]. Estes são apenas alguns exemplos de ABs. Cada um deles é baseado em princípios observados na natureza, como evolução, comportamento de enxame, interações sociais entre animais, entre outros.

Muitos problemas complexos de otimização em engenharia, logística e biologia não podem ser resolvidos por métodos exatos de otimização em tempo polinomial. Para esta classe de problemas, os chamados problemas $\mathcal{NP}$-difíceis, os ABs são frequentemente utilizados para encontrar soluções subótimais. Além disso, os Modelos de Ilhas Paralelas (MIPs) têm recebido considerável atenção, pois são capazes de utilizar ABs para melhorar a busca por soluções novas e não convencionais (*exploração*) e aprofundar a exploração de áreas conhecidas em busca de soluções de alta qualidade (*aproveitamento*[1]), resultando em melhorias de aceleração (*speedup*) e acurácia. (por exemplo, [9], [10], [11]). Os MIPs compreendem um conjunto de ilhas, cada uma contendo instâncias de ABs, evoluindo suas próprias populações simultaneamente. Periodicamente, soluções candidatas são trocadas entre as ilhas por meio de um processo chamado migração. Essa transferência de soluções está sujeita a um conjunto de regras conhecidas como política de migração e envolve parâmetros que requerem uma calibração cuidadosa para obter resultados satisfatórios. As topologias de comunicação desempenham um papel importante nos parâmetros de migração e podem ser classificadas como estáticas ou dinâmicas. As topologias estáticas estabelecem conexões predefinidas entre as ilhas, formando vizinhanças fixas durante o processo de migração. Por outro lado, as topologias dinâmicas permitem a reconfiguração de conexões ao longo do ciclo evolutivo, possibilitando variações nas vizinhanças durante o processo de migração.

A abordagem dos MIPs expande a resolução de problemas $\mathcal{NP}$-difíceis, oferecendo

[1]Usamos "exploração" e "aproveitamento" para os conceitos de *exploration* and *exploitation*.

uma maneira de lidar com a complexidade desses problemas desafiadores. Ao emular os princípios da evolução natural, esses modelos fornecem uma alternativa eficaz para buscar soluções de alta qualidade, fomentando uma exploração diversificada do espaço de busca. Com sua adaptabilidade e capacidade de encontrar soluções subótimas viáveis, os MIPs continuam desempenhando um papel importante na pesquisa em ciência da computação, abrindo caminho para aprimorar a acurácia das soluções para problemas complexos com um ganho notável de aceleração. Este documento é dedicado a explorar a aplicação e os benefícios dos MIPs como objeto central de estudo.

1.5 Motivação

Resolver problemas $\mathcal{NP}$-Difíceis requer técnicas algorítmicas que não permitem o cálculo de soluções exatas em tempo polinomial e têm sido objeto de estudo em ciência da computação desde antes da década de 1970, quando as noções de complexidade computacional foram consolidadas. Assim, temos que as soluções que podemos encontrar para tais problemas em tempo polinomial são aproximadas. Os problemas clássicos, como o problema da Mochila, Conjunto Independente, Coloração de Grafos, Satisfazibilidade Booleana e Roteamento de Veículos, são amplamente estudados na literatura e têm aplicações em vários campos, incluindo logística, seleção de características, pesquisa operacional e ciência da computação, sendo utilizados como problemas de referência para testar e comparar a eficiência de diferentes algoritmos e técnicas de otimização. Devido à natureza adaptativa simples dos ABs, nós recorremos a uma variedade desses algoritmos para calcular soluções de boa qualidade para tais problemas.

A evolução da tecnologia de hardware tem possibilitado cada vez mais o uso de algoritmos paralelos. Os ABs têm atraído grande interesse de pesquisadores que têm explorado uma variedade de algoritmos paralelos buscando melhorar a qualidade das soluções (sequenciais). A simplicidade de modelagem paralela dessa classe de algoritmos permitiu uma abordagem mais eficiente na exploração de soluções para problemas complexos. Nesse contexto, uma maneira simples e eficaz de trabalhar com ABs é por meio de MIPs, que possibilitam explorar várias soluções em paralelo e colaboração entre ABs, aumentando a precisão e, é claro, a velocidade.

No contexto de MIPs, dois enfoques fundamentais emergem: MIPs homogêneos e MIPs heterogêneos, resumidamente HoMIPs e HeMIPs, respectivamente. Inspirados na analogia de ilhas isoladas dentro de um arquipélago, esses enfoques compartilham a ideia de que cada ilha representa um conjunto de soluções potenciais para o problema dado. No entanto, diferem significativamente em suas características e abordagens. Nos HeMIPs, as ilhas possuem características distintas, potencialmente incorporando uma variedade de técnicas de busca e ABs. Este enfoque valoriza a diversidade, enriquecendo a exploração do espaço de busca e possibilitando diferentes estratégias para colaborar na busca por soluções de alta qualidade. Por outro lado, os HoMIPs visam uniformidade em todas as ilhas, empregando apenas um AB e estratégias idênticas para explorar o espaço de busca. Isso promove uma exploração coesa e uma troca eficaz de informações entre as ilhas.

Um componente crítico para o sucesso tanto dos HoMIPs quanto dos HeMIPs é a migração entre as ilhas. Os parâmetros de migração determinam a frequência e quais soluções são escolhidas para migrar. A manipulação habilidosa desses parâmetros é essencial para alcançar soluções de qualidade. Portanto, a seleção cuidadosa e ajuste dos parâmetros de migração desempenham um papel crucial na busca por soluções de qualidade em MIPs, seja por meio de abordagens HoMIPs ou HeMIPs. O equilíbrio entre a exploração global do espaço de busca e a exploração local detalhada é alcançado por meio desses parâmetros, moldando a eficácia e o desempenho geral da estratégia de otimização. Além disso, a eficácia é influenciada pela seleção cuidadosa do AB usado para abordar o problema específico, bem como pelas estratégias empregadas para evoluir a população global.

1.6 Contribuição

A contribuição deste trabalho abrange uma análise minuciosa dos modelos de ilhas paralelas, considerando cuidadosamente as complexidades dos mecanismos que devem ser gerenciados para extrair soluções aprimoradas dessa metodologia.

A pesquisa sobre MIPs começou com a implementação de um MIP Homogêneo (HoMIP) com migração e seu contraponto sem migração em [12]. Neste último modelo, cada ilha evolui independentemente e, no final, o indivíduo mais apto entre todas as ilhas é considerado a solução. No entanto, os resultados revelaram soluções de baixa qualidade no HoMIP com migração, inferiores à versão sequencial, o que motivou uma investigação sobre esse desempenho. Esta fase inicial de pesquisa precedeu os experimentos realizados em [13], onde a calibração de parâmetros foi introduzida para melhorar significativamente a qualidade das soluções fornecidas pelos HoMIPS, superando a precisão fornecida pela versão sequencial. Vale notar que, embora uma abordagem de calibração de parâmetros tenha sido adotada, a metodologia empregada não foi complexa. Um método ganancioso foi escolhido para determinar valores de parâmetros que pudessem fornecer soluções aprimoradas. Esta escolha estratégica enfatiza a importância de ajustar os parâmetros evolutivos e migratórios, preenchendo uma lacuna na literatura que muitas vezes negligenciava a importância desses valores na busca por soluções de qualidade. Estabelecer a necessidade de uma abordagem proativa de calibração de parâmetros contribui substancialmente para o avanço da pesquisa em MIPs. De fato, conforme evidenciado em [14, 15, 16], a calibração de parâmetros é essencial no desenvolvimento de MIPs Heterogêneos (HeMIPs), que melhoram as soluções.

As topologias de comunicação entre ilhas são importantes para determinar como a migração pode ocorrer, seja estática ou dinamicamente, e impactam diretamente a qualidade da solução ao indicar quais ilhas estão envolvidas no processo de migração. Em [17], uma topologia dinâmica foi introduzida, na qual a migração pode ser ajustada dinamicamente com base na qualidade das soluções e na diversidade encontrada entre as ilhas durante a evolução. A topologia estabelece a estrutura de comunicação entre as ilhas de acordo com seu comportamento evolutivo, influenciando a eficácia do processo de migração e, consequentemente, afetando a qualidade geral das soluções alcançadas. A proposta de uma topologia dinâmica representa um avanço em comparação com as topologias estáticas convencionais, conforme discutido em [18, 19, 15, 20, 16].

As populações iniciais nas ilhas, conforme evidenciado em [13, 17], foram um aspecto crucial da pesquisa. Os experimentos demonstram que o uso de populações iniciais a partir da versão sequencial leva a soluções mais robustas e eficientes nos HoMIPs. Esta descoberta melhorou significativamente a qualidade das soluções obtidas e foi adotada em todos os trabalhos subsequentes.

A análise comparativa entre HoMIPs síncronos e assíncronos, discutida em [16], revelou uma dependência crítica dos parâmetros específicos de migração para cada problema. A sensibilidade desses parâmetros foi significativamente destacada, enfatizando sua influência direta na eficiência dos HoMIPs. Isso ressalta a importância crucial de calibrar esses parâmetros, como observado em [12], pois os experimentos mostraram que o mesmo modelo, seja síncrono ou assíncrono, apresenta comportamentos distintos tanto em precisão quanto em aceleração. Essas observações se estendem aos modelos heterogêneos, conforme discutido em [14].

Explorar vários algoritmos evolutivos em [19] forneceu discernimento valioso para a implementação de HeMIPs capazes de fornecer resultados competitivos. Contrariando a visão predominante na literatura, foram identificados cenários em que os HoMIPs superam os HeMIPs em precisão [14]. Para melhorar a precisão dos HeMIPs, focamos no desenvolvimento da reconfiguração de algoritmos de ilhas. Essa metodologia permite que as ilhas executem diferentes algoritmos durante a evolução [15] oferece maior dinamismo e flexibilidade, onde, durante um intervalo fixo de evolução, a ilha com o pior desempenho passa por um processo no qual o algoritmo evolutivo é substituído pelo algoritmo da ilha que mostrou a melhor evolução em sua população. Em uma etapa subsequente de pesquisa, a metodologia de reconfiguração foi refinada adotando a reconfiguração de ilhas baseada em estagnação [16]. O conceito de estagnação está ligado à melhoria da aptidão do melhor indivíduo. Esta estratégia de reconfiguração tornou-se um processo dinâmico altamente flexível, concedendo a cada ilha autonomia para acionar o processo de reconfiguração sempre que sua população estagnar. Ao contrário da proposta inicial de uma única reconfiguração em intervalos fixos em [15], a reconfiguração baseada em estagnação abre a porta para múltiplos processos de reconfiguração. Esta abordagem resultou em soluções aprimoradas em comparação com a abordagem proposta em [15] e demonstrou superioridade sobre todas as versões anteriormente propostas de HoMIPs e HeMIPs.

As abordagens utilizadas para analisar problemas difíceis de otimização consistem em aplicar hiper-heurísticas. Tais métodos são baseados na seleção ou geração, em um espaço de busca de heurísticas, das (meta)-heurísticas mais adequadas para resolver problemas de otimização, em vez de buscar apenas no espaço de solução do próprio problema. Em uma pesquisa recente, Dokeroglu et al. [21] examinam os trabalhos mais citados sobre hiper-heurísticas publicados nos últimos vinte anos em revistas e conferências relevantes na área. Os autores classificam hiper-heurísticas em quatro classes, não necessariamente disjuntas: hiper-heurísticas de seleção, heurísticas de baixo nível, problemas de otimização-alvo e hiper-heurísticas paralelas. Considerando essa classificação, nosso trabalho pode ser visto como uma heurística de baixo nível, pois combina algoritmos bioinspirados e algoritmos exatos dentro de modelos de ilhas para calcular a aptidão dos indivíduos para resolver problemas de otimização. Além disso, pode ser visto como uma hiper-heurística paralela, restrita ao paralelismo em nível algorítmico, onde diferentes (meta)-heurísticas são executadas independentemente ou cooperativamente em paralelo.

Além disso, a taxonomia genérica de otimização inspirada na natureza com múltiplas populações inclui a estrutura dos MIPs (por exemplo, Ha et al. [22]). Essencialmente, a taxonomia genérica de múltiplas populações refere-se a abordagens de otimização que dividem a população em pequenas subpopulações realizando operações evolutivas, como seleção, cruzamento e mutação para evoluir indivíduos, permitindo a interação dessas subpopulações por meio de processos de fusão, comunicação e redivisão. Os objetivos, como esperado, são evitar a convergência prematura e manter a diversidade da população para garantir soluções de alta qualidade. A experiência obtida no desenho e aplicação de algoritmos de múltiplas populações, geralmente descritos usando termos como “paralelos”, “cooperativos”, “co-evolução”, “ilhas”, entre outros [22], é relevante para os propósitos vitais em nosso estudo. Questões críticas apresentadas em algoritmos de múltiplas populações estão relacionadas a várias questões que surgiram no desenvolvimento de nossa pesquisa. Por exemplo, abordar a questão de como determinar o número de subpopulações nos permitiu projetar MIPs com um número equilibrado de ilhas e tamanhos de populações de ilhas; explorar como gerenciar a comunicação entre subpopulações nos permitiu desenvolver topologias de comunicação e políticas de migração eficazes; finalmente, pesquisar minuciosamente como definir a estratégia de busca de cada subpopulação nos levou a investigar MIPs heterogêneos e reconfiguração dinâmica, que, até onde sabemos, não foram abordados em trabalhos anteriores. Além disso, abordar outra questão que surge em algoritmos de múltiplas populações, como determinar a área de busca de cada subpopulação, é de grande interesse para pesquisas futuras. Abordar essa questão é importante para determinar uma distribuição eficiente da exploração do espaço de busca pelas populações das ilhas, reduzindo a redundância individual e promovendo uma exploração abrangente do espaço de busca, evitando assim a convergência para ótimos locais.

Além das duas taxonomias mencionadas acima (hiper-heurísticas e modelos de múltiplas populações), o Capítulo 5 sobre trabalhos relacionados apresenta referências mais precisas comparando estas com o nosso trabalho.

1.7 Estrutura do Trabalho

O documento está estruturado nos capítulos descritos abaixo.

Capítulo 2 [Referencial Teórico] apresenta o contexto relacionado a ABs, MIPs e problemas de referência.

Capítulo 3 [Visão Geral da Evolução da Pesquisa] descreve o trabalho conduzido durante a pesquisa, culminando no desenvolvimento de uma nova metodologia para modelos de ilha baseados em heterogeneidade e reconfiguração.

Capítulo 4 [MIPs Reconfiguráveis com Política de Reconfiguração Baseada em Estagnação] discute os HeMIPs com reconfiguração baseada em estagnação.

Capítulo 5 [Trabalho Relacionado] revisa trabalhos relacionados.

Capítulo 6 [Conclusão] conclui e discute possíveis trabalhos futuros.

1.8 Glossary

$\simeq$	Dynamic Complete Graph Topology with migration between islands of the same performance
`12A 12S`	12-island asynchronous and synchronous
`24A 24S`	24-island asynchronous and synchronous
BA	Bio-inspired Algorithm (AB in Portuguese)
BACI	BA Classification Interval
CEC	Congress on Evolutionary Computation
CSA	Crow Search Algorithm
CSA	Cuckoo Search Algorithm
DE	Differential Evolution
DMOP	Dynamic Multi-Objective Optimization Problem
EP	Emigration Policy
EMI	Emigrating Individuals
F_M	DE mutation factor
GA	Genetic Algorithm
GAD	Double-point Crossover GA
`gbmm`	Dynamic Complete Graph Topology with migration between god-bad and medium-medium islands
GWO	Grey Wolf Optimization Algorithm
HePIM	Heterogeneous PIM (HeMIP in Portuguese)
HoPIM	Homogeneous PIM (HoMIP in Portuguese)
IMI	Immigrating Individuals
IN	Individuals Number
MI	Migration Interval
P_C	DE probability of crossover
PIM	Parallel Island Model (MIP in Portuguese)
PSO	Particle Swarm Optimization

$\mathcal{P}^{\texttt{GA}}_{\text{Tr12A}}$, $\mathcal{P}^{\texttt{GAD}}_{\text{Tr12A}}$, $\mathcal{P}^{\texttt{DE}}_{\text{Tr12A}}$, $\mathcal{P}^{\texttt{PSO}}_{\text{Tr12A}}$	`GA`, `GAD`, `DE` and `PSO` based asynchronous static 12-island tree HoPIMs
$\mathcal{P}^{\texttt{GA}}_{\text{Tr12S}}$, $\mathcal{P}^{\texttt{GAD}}_{\text{Tr12S}}$, $\mathcal{P}^{\texttt{DE}}_{\text{Tr12S}}$, $\mathcal{P}^{\texttt{PSO}}_{\text{Tr12S}}$	`GA`, `GAD`, `DE` and `PSO` based synchronous static 12-island tree HoPIMs
$\mathcal{P}^{\texttt{GA}}_{\text{gbmm12A}}$, $\mathcal{P}^{\texttt{GAD}}_{\text{gbmm12A}}$, $\mathcal{P}^{\texttt{DE}}_{\text{gbmm12A}}$, $\mathcal{P}^{\texttt{PSO}}_{\text{gbmm12A}}$	`GA`, `GAD`, `DE` and `PSO` based asynchronous dynamic 12-island complete graph HoPIMs
$\mathcal{P}^{\texttt{GA}}_{\text{gbmm12S}}$, $\mathcal{P}^{\texttt{GAD}}_{\text{gbmm12S}}$, $\mathcal{P}^{\texttt{DE}}_{\text{gbmm12S}}$, $\mathcal{P}^{\texttt{PSO}}_{\text{gbmm12S}}$	`GA`, `GAD`, `DE` and `PSO` based synchronous dynamic 12-island complete graph HoPIMs
$\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$, $\mathcal{P}^{\text{Het}}_{\text{Tr12S}}$	asynchronous and synchronous static 12-island tree HePIM
$\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$, $\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$	asynchronous and synchronous dynamic 12-island complete graph HePIM
$\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$, $\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$	static 12-island tree reconfigurable and stagnation-based reconfigurable HePIMs
$\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$, $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$	dynamic 12-island complete graph reconfigurable and stagnation-based reconfigurable HePIMs
RF	Reconfiguration Frequency
SSA	Social Spider Algorithm
SSCI	Symposium Series on Computational Intelligence
SRD	Signed Reversal Distance
STD	Signed Translocation Distance
TMP	Task Mapping Problem
`Tr`	Static Tree Topology
URD	Unsigned Reversal Distance
UTD	Unsigned Translocation Distance
WOA	Whale Optimization Algorithm

Chapter 2

Theoretical Referential

The chapter is organized as follows. In Section 2.1, the selected BAs, which serve as the backbone of the experiments, are presented, discussing their characteristics. Next, in Section 2.2, the understanding of parallel island models is deepened, exploring the essential concepts and terminologies fundamental to understanding the applied methodology. Section 2.3 highlights the benchmark problems used as case studies to validate the research.

2.1 Bio-inspired Algorithms

The BAs chosen in the experiments are Genetic Algorithms (`GA`), Self-Adjusting Particle Swarm Optimization (`PSO`), Social Spider Algorithm (`SSA`), and Differential Evolution (`DE`). `GA` was selected because, in previous work [23, 12, 13], PIMs provided excellent solutions for two case studies that are used in this research: Unsigned Translocation Distance and Unsigned Reversal Distance, both detailed in Section 2.3. On its side, `PSO` is a well-known, extensively applied optimization algorithm. Also, `DE` is frequently used to address problems related to those of the current research ([24, 25, 26]). Finally, `SSA` being a more recent algorithm, offers modern optimization approaches. Additionally, it is essential to emphasize that the PIMs are not dependent on these specific algorithms, making them applicable to any BA.

The GA is inspired by the Darwinian principle of species evolution [27]. GA is a probabilistic method that evolves through reproduction with an adaptive search engine based on the survival principle of the fittest. In this work, the GA is used with a breeding cycle composed of the following operators: *selection*, *crossover*, *mutation*, and *replacement*. These operators are controlled by adjustable parameters within the algorithm, allowing for fine-tuning and customization.

This algorithm evolves a population by considering a breeding cycle where the best individuals are selected to produce offspring by applying one-point crossover (Figure 2.1 (a)). Then, the descendants replace the worst individuals in the current population. The mutation process is conservative. A low mutation rate is applied (as naturally occurs in organisms) since exaggerated mutations corrupt inherited genes. In addition to incorporating the basic GA into the experiments, it was chosen to employ the Double-point Crossover Genetic Algorithm (GAD). The algorithm GAD is a variant from the GA in which the *crossover* uses the technique illustrated in Fig. 2.1 (b). In this technique, the *replacement* of individuals by descendants in the current population randomly selects the individuals to be replaced. As shown in [28], the results and behavior of both algorithms are quite different. The flowcharts for algorithms GA and GAD are illustrated in Figure 2.2.

We use GA in research involving HoPIMs from works [12, 13, 17, 18, 19, 20], and it is also employed in HePIMs [14, 15, 16]. On the other hand, GAD is exclusively applied in research related to HePIMs [14, 15, 16].

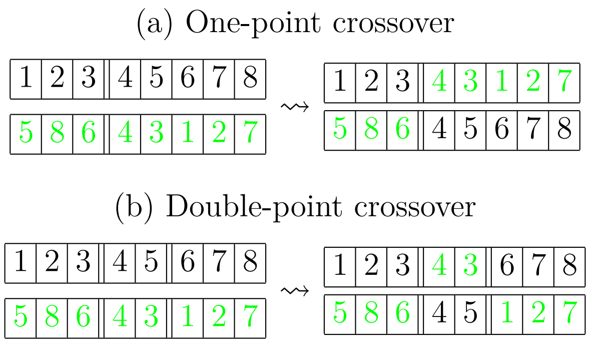


Figure 2.1: One-point and double-point crossing operators.

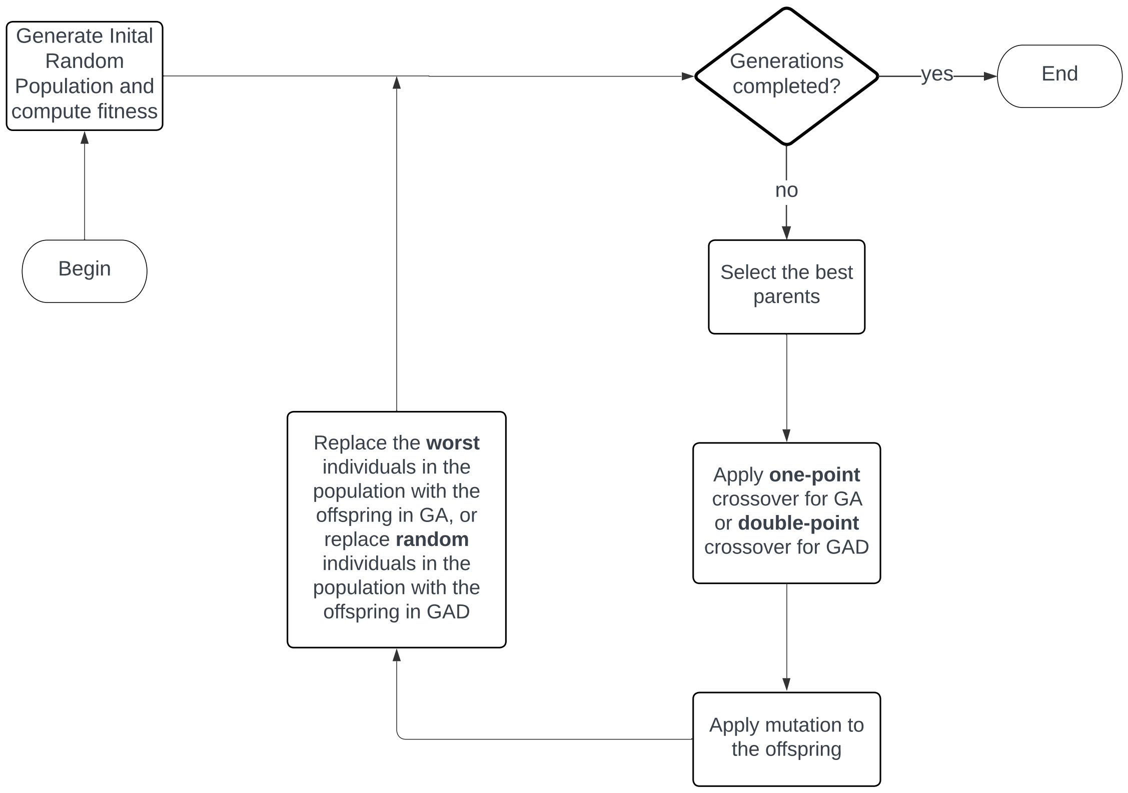

Generate Inital Random Population and compute fitness
Generations completed?
yes
End
no
Begin
Select the best parents
Replace the **worst** individuals in the population with the offspring in GA, or replace **random** individuals in the population with the offspring in GAD
Apply **one-point** crossover for GA or **double-point** crossover for GAD
Apply mutation to the offspring

Figure 2.2: Flowchart of the GA and GAD.

2.1.1 Self-Adjusting Particle Swarm Optimization

Particle Swarm Optimization (PSO) was introduced in [29] to address continuous domain problems inspired by the behavior of social organisms in groups, such as species of birds and schools of fish. In PSO each particle, $x_i = (x_{i1}, x_{i2}, \cdots, x_{ij}, \cdots, x_{in})^{\mathrm{T}}$, represents a point in the search space and its current position in an n-dimensional space. Variables x_{ij} denote the vector component of the i-th particle in the j-th dimension. The motion of a i-th particle is specified by the velocity vector $v_i = (v_{i1}, v_{i2}, \cdots, v_{ij}, \cdots, v_{in})^{\mathrm{T}}$ with better position of the i-th particle being stored in *pbest*. The algorithm maintains the best position for all particles denoted as *gbest*, which guides the particles in the search space. The velocity vector v_{ij}^{k+1} and position $x^{(i,k+1)}$ for the i–th particle in the step $k+1$

iteration is given by the expression:

$$\begin{aligned} v^{(i,k+1)} &= w.v^{(i,k)} + c_1.rand_{1ij}.(pbest^k_{ij} - x^k_{i,j}) + c_2.rand_{2ij}.(gbest^k_{ij} - x^k_{i,j}) \\ x^{(i,k+1)} &= x^{(i,k)} + v^{(i,k+1)} \end{aligned} \tag{2.1}$$

In Eq. (2.1), w is the weight of inertia (momentum) that introduces friction into the particles' motion, reducing inertial velocity; c_1 and c_2 are individual and global acceleration coefficients that influence the maximum step size a particle can take and, $rand_{1ij}$ and $rand_{2ij}$ are vectors containing randomly uniformly distributed numbers generated at each iteration k, which values range from 0 to 1.

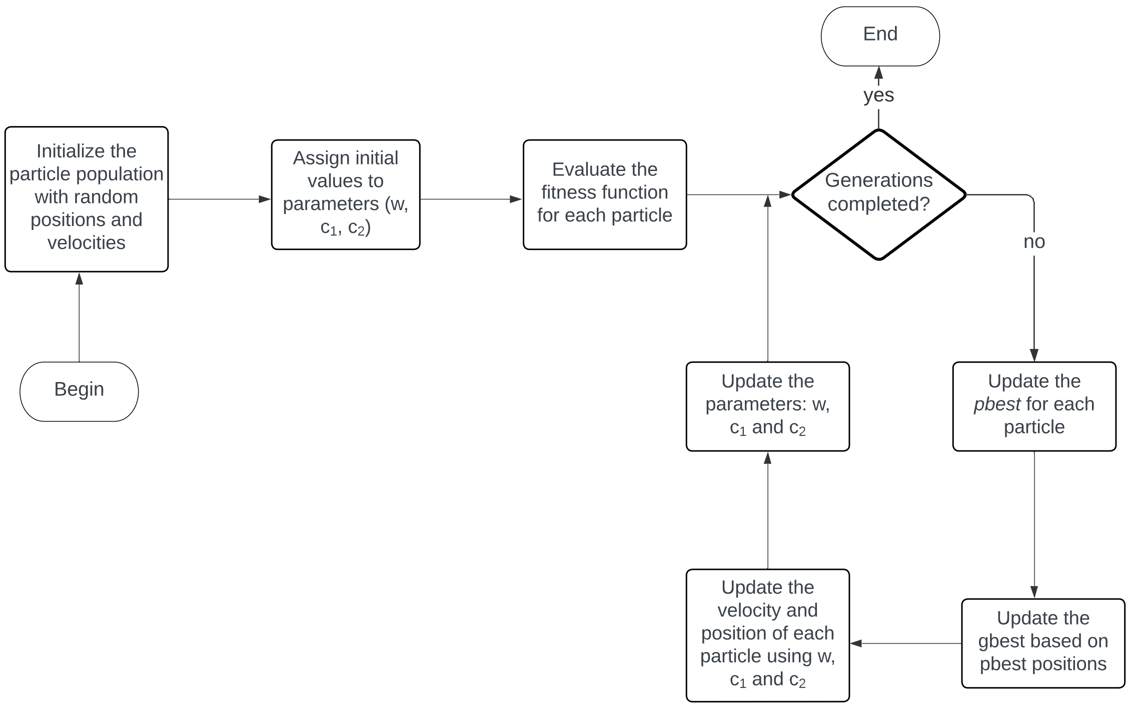
Adaptability in an optimization algorithm can be defined as self-tuning according to rules that have variable parameters. This work uses the self-adaptive PSO proposed in [30], where the parameters w, c_1, and c_2 and the update frequency of each particle are used to perform the self-tuning in the search process, adjusting each particle through the equation:

$$\begin{aligned} w_i^{k+1} &= w_i^k + \alpha_i^k \cdot (w^k_{\text{best}} - w_i^k), \ : i = 1, 2, \cdots, m \\ c_{1i}^{k+1} &= c_{1i}^k + \alpha_i^k \cdot (c^k_{1\text{best}} - c_{1i}^k), \ : i = 1, 2, \cdots, m \\ c_{2i}^{k+1} &= c_{2i}^k + \alpha_i^k \cdot (c^k_{2\text{best}} - c_{2i}^k), \ : i = 1, 2, \cdots, m \end{aligned} \tag{2.2}$$

We apply PSO in our research encompassing HoPIMs in [19] and HePIMs in [14, 15]. The flowchart for PSO is illustrated in Figure 2.3.

2.1.2 Social Spider Algorithm

The locomotion of spiders has been a subject of research in bionic engineering, particularly in the context of designing robots [31], with a specific focus on social spiders. These spiders, such as *Mallos gregalis*, live in groups and interact with other spiders in the same group. Based on social spiders, a method for solving optimization problems has been designed in [32].



End
yes
Initialize the particle population with random positions and velocities
Assign initial values to parameters (w, c_1, c_2)
Evaluate the fitness function for each particle
Generations completed?
no
Begin
Update the parameters: w, c_1 and c_2
Update the *pbest* for each particle
Update the velocity and position of each particle using w, c_1 and c_2
Update the gbest based on pbest positions

Figure 2.3: Flowchart of the PSO.

In the Social Spider Algorithm (SSA), the optimization problem search space is formulated as a hyperdimensional spider web. A position on the web represents an acceptable solution to the problem, and all solutions to the problem have corresponding positions on that web. In addition to mapping solutions to the problem, the web transmits spider-generated vibrations. Each spider on the web holds a position, and the solution's quality is based on the objective function representing the potential to find a food source in the position.

Spiders can explore every search space mapped on the web. However, spiders cannot leave the web since off-web positions represent unfeasible solutions to the problem. When a spider moves to a new position, a vibration is generated and propagated through the net. A vibration contains information from the propagating spider that will eventually be consulted by other spiders for food source information.

A set of information is required on each spider in the SSA: 1) spider position on the web; 2) spider fitness; 3) The vibration of the spider in the previous iteration; 4) the number of iterations since spider changed its vibration; 5) the movement that the spider performed in the previous iteration; 6) dimension mask represented by a binary vector $(0-1)$ with size equal to the input that spider employed to guide movement in the previous iteration.

The three phases in `SSA` are described below.

- Initialization: the algorithm defines the objective function and its solution space. The parameter value used in `SSA`: vibration rate (p_v), probability of changing mask (p_c), and percentage to control updates to mask positions (p_m) are assigned. After setting the values, the algorithm creates an initial optimization spider population that remains unchanged during the `SSA` simulation. Spider positions with their fitness values are randomly generated on the search space. The initial target vibration of each spider in the population is set at its current position, and the vibration intensity is assigned to be zero.

- Iteration: in each iteration, all spiders on the web go through the following steps: fitness assessment, vibration generation, mask change, and random walk. The algorithm calculates the fitness of all spiders once and updates the overall ideal value. Then, these spiders generate vibrations in their positions and propagate such vibrations in the web. In this process, each spider receives vibrations from all other spiders. Information on these vibrations includes the source position of the vibration and its attenuation intensity. After receiving all the vibrations, the current spider takes the strongest vibration called v_{best} and compares it with its current vibration v_s^{local}; if v_s^{best} is higher, v_s^{local} updates the value by the value in v_s^{best} and the number since the current spider vibration was changed denoted as c_s is set to zero; otherwise c_s is incremented by 1 and the value in v_s^{local} is kept. Subsequently, the current spider performs a random walk toward v_s^{local} using the mask dimension to guide the movements. Each spider maintains its mask, where initially, all values are zero. In each iteration, the current spider has a probability of $1 - {p_c}^{c_s}$ to change its mask with $p_c \in (0,1)$. If the mask changes, each bit mask has a p_m probability of being changed.

- Stopping criteria: maximum number of iterations (used in this work).

We employed the SSA in HoPIMs in work [19]. Figure 2.4 illustrates the flowchart of SSA.

2.1.3 Differential Evolution Algorithm

The Differential Evolution (DE) algorithm is a stochastic optimization method that was proposed in [3]. The BA was designed to tackle global optimization problems involving highly nonlinear functions and complex search spaces. The algorithm uses NP vectors with D dimensions to represent individuals in the population for each generation G.

$$x_{i,G}, i = 1, 2, \cdots, NP \tag{2.3}$$

The initial population is randomly chosen and evolves by adding the weighted difference between two population vectors to a third vector. This operation is called mutation.

For each target vector $x_{i,G}, i = 1, 2, \cdots, NP$, a mutant vector is generated as follows:

$$v_{i,G+1} = x_{r_1,G} + F_M * (x_{r_2,G} - x_{r_3,G}) \tag{2.4}$$

r_1, r_2, r_3 are different vectors randomly selected from the current population, and $F_M > 0$. Due to the nature of the mutation, the authors suggest working with F_M in the range $[0, 2]$. Subsequently, a crossover operator called P_C, which represents the probability of applying crossover, is used to improve diversity. Thus, the trial vector: $v_{i,G+1} = (v_{1i,G+1}, v_{2i,G+1}, \cdots, v_{Di,G+1})$ is formed as follows:

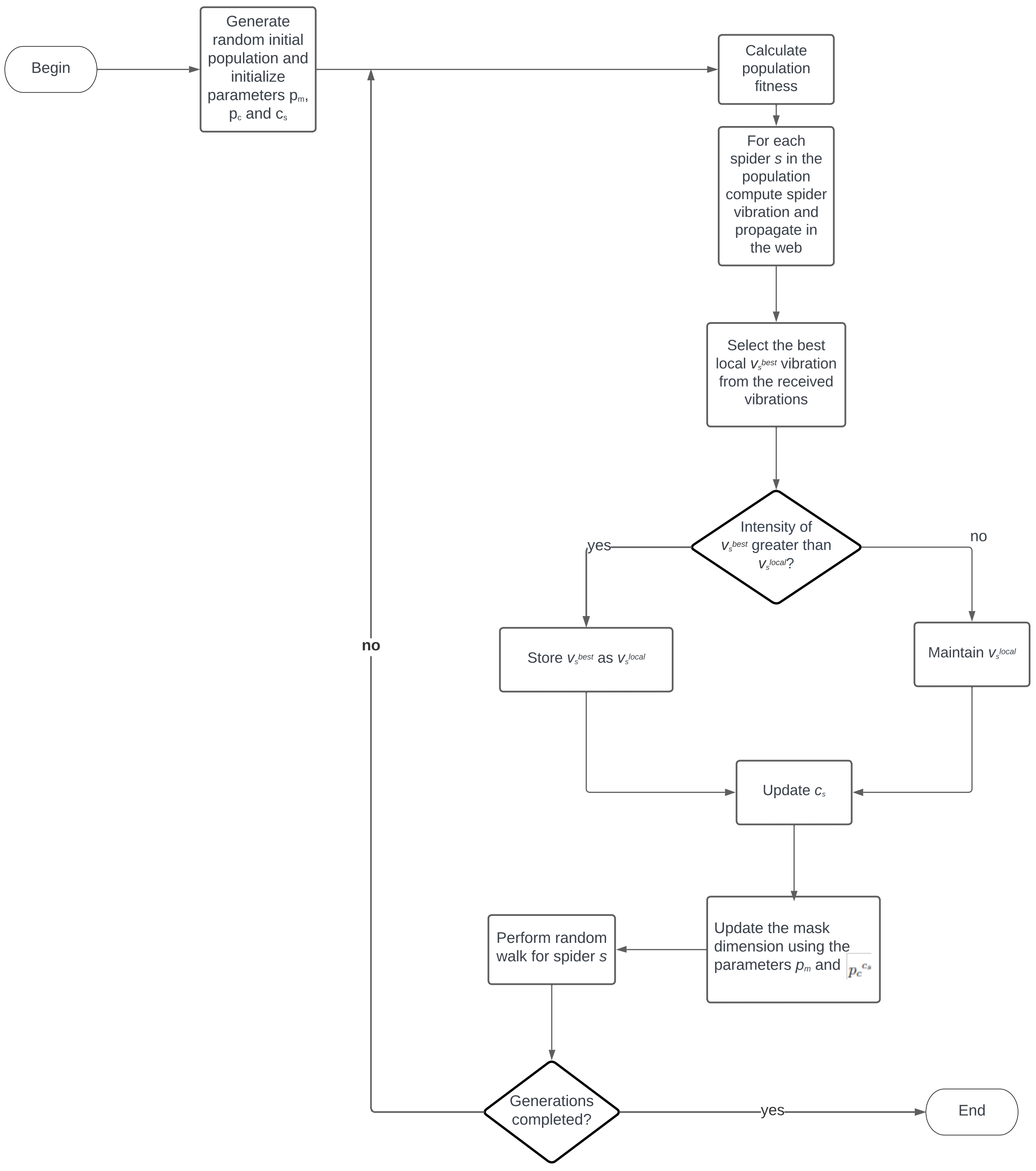

Begin
Generate random initial population and initialize parameters p_m, p_c and c_s
Calculate population fitness
For each spider s in the population compute spider vibration and propagate in the web
Select the best local v_s^best vibration from the received vibrations
Intensity of v_s^best greater than v_s^local?
yes
no
Store v_s^best as v_s^local
Maintain v_s^local
Update c_s
Update the mask dimension using the parameters p_m and p_c^c_s
Perform random walk for spider s
Generations completed?
no
yes
End

Figure 2.4: Flowchart of the SSA.

$$u_{ji,G+1} = \left\{ \begin{array}{llll} v_{ji,G+1} & if & (randb(j) \leq P_C) & or \quad j = rnbr(i) \\ x_{ji,G} & if & (randb(b) > P_C) & and \quad j \neq rnbr(i) \end{array} \right\} \qquad j = 1, 2, \cdots, D. \tag{2.5}$$

$randb(j)$ represents the $j-th$ evaluation of a randomly generated number within the range $\in [0,1]$. P_C is the crossover parameter value $\in [0,1]$. Meanwhile, $rnbr(i)$ returns the value of the vector at the corresponding index of the mutant vector $v_{i,G+1}$.

In the population update step, the trial vector $u_{i,G+1}$ is compared to the vector $x_{i,G}$. If the fitness function of the trial vector is better than that of vector $x_{i,G}$, then the trial vector will take its place in the next generation; otherwise, the trial vector is discarded.

The `DE` is used in [14, 15, 16], which is applied to HePIMs. The corresponding flowchart is illustrated in Figure 2.5.

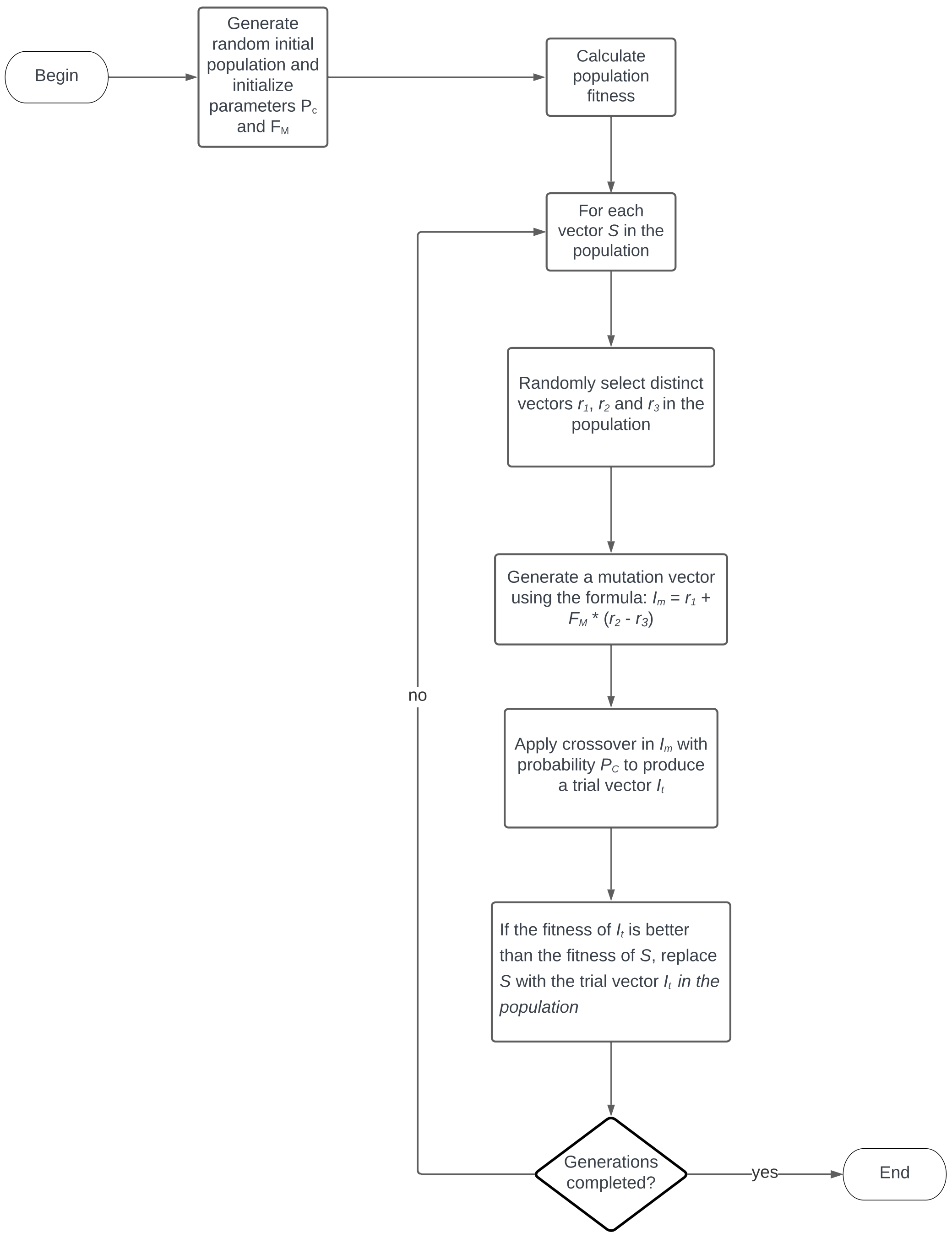
2.1.4 Other algorithms

In addition to addressing the algorithms explored in this work, it is pertinent to emphasize that the field of bioinspired algorithms is continually expanding and being incorporated into model islands. Among these algorithms, we have the Cuckoo Search Algorithm (CSA) [33], the Grey Wolf Optimization Algorithm (GWO) [34], the Whale Optimization Algorithm (WOA) [35], and the Crow Search Algorithm (CSA) [36].

The CSA is a metaheuristic inspired by the intelligent foraging behavior of cuckoo birds. It employs parasitism strategies, where cuckoos lay eggs in the nests of other birds, representing candidate solutions. The algorithm aims to optimize functions by preserving suitable solutions and exploring new areas of the search space.

The social hierarchy of wolf packs inspires the GWO. This bioinspired algorithm simulates the social dynamics of grey wolves, including alpha leaders and subordinate members. Wolves collaborate to hunt prey, representing function optimization. GWO seeks to find the best solution by balancing exploration and exploitation.

The cooperative hunting behavior of whales inspires the WOA. It introduces the concept of whales coordinating to encircle prey, representing solutions in a search space. WOA aims to optimize functions using strategies to update whale positions and find high-quality solutions.



Begin
Generate random initial population and initialize parameters P_c and F_M
Calculate population fitness
For each vector S in the population
Randomly select distinct vectors r_1, r_2 and r_3 in the population
Generate a mutation vector using the formula: $I_m = r_1 + F_M * (r_2 - r_3)$
Apply crossover in I_m with probability P_C to produce a trial vector I_t
If the fitness of I_t is better than the fitness of S, replace S with the trial vector I_t *in the population*
Generations completed?
no
yes
End

Figure 2.5: Flowchart of the DE.

The CSA, inspired by the intelligent behavior of crows, is a population-based technique. Crows, widely recognized as some of the most intelligent animals, exhibit search behavior akin to an optimization process, hiding and retrieving food in specific locations within their environment. From an optimization perspective, crows act as seekers; the environment represents the search space, each position in the environment corresponds to a viable solution, the quality of the food source is the objective function, and the best food source in the environment equates to the global solution of the problem. The CSA aims to emulate this intelligent behavior of crows to find solutions to optimization problems.

In Section 3.3, we will address studies exploring the application of CSA, GWO, WOA, and CSA in island models.

2.2 Parallel Island Models

Studies involving the Parallel Island Models (PIMs) emerged in the 1980s with Pettey et al. in [37].

Due to the abundance of studies demonstrating improved solutions using PIMs, as evidenced in the literature review presented in Chapter 5, we have chosen to focus our research on exploring this model. Furthermore, it is essential to note that the concept of PIMs proves to be versatile and can be applied to any BA([24],[38],[39],[40],[41],[42],[43],[44]).

An important point is that conducting independent runs for extended periods may not be advantageous in PIMs. This aspect occurs because individual islands can become stuck in optimal locations, which is not ideal, as other islands could simultaneously explore the most promising regions within the search space. Therefore, establishing communication between islands to coordinate research efforts becomes viable when islands are trapped in optimal locations or experience low-quality outcomes.

In PIMs, each island maintains an independent population that evolves autonomously. Periodically, individuals are shared between islands through a process known as migration. An organizational architecture, referred to in the literature as the migration topology, defines a communication model between islands. Individuals are selected from the local island at specific intervals and sent to target islands (those that receive individuals from a local island). These individuals are incorporated according to particular policies. Below, we highlight each policy:

Number of immigrants and emigrants the number of individuals who will be sent and received from one island to another. One might select the best, worst, or random individuals.

Emigration policy remove or clone individuals, considering the choice among worse, random, or better individuals.

Immigration policy replace individuals considering the choice among worse, random, or better individuals on the target island.

Migration interval number of generations between one migration and another.

Migration topology organizational model that defines which islands will communicate with which islands.

Synchronization policy Migration between islands can be asynchronous or synchronous. When synchronizing the migration, the islands evolve at approximately the same rate (number of generations, percentage of convergence, etc.), determining when a migration should occur. Asynchronous migration is unrelated to the state of evolution in all the islands. This asynchronous behavior is typically found in nature since different environments are responsible for differences in the speed of evolution.

Parameters involved in Migration Policies

The migration policies deliberated in the previous section are essential for shaping the performance and accuracy of PIMs. These policies encompass parameters that, when properly defined, significantly enhance the likelihood of improving solutions for complex problems within a reasonable timeframe. Conversely, an inadequate definition can detrimentally affect the quality of solutions to the extent that they may be inferior to their sequential counterparts. The explanation for each parameter is as follows:

- *NumMigIndividuals* represents the number of individuals for migration.
- *TypeEmIndividual* represents the type of individuals selected for emigration among:
    1. Better individuals.
    2. Worse individuals.
    3. Random individuals.
- *EmPolicy* represents the emigration policy:
    1. Clone individuals.
    2. Remove individuals.
- *TypeImIndividual* represents the immigration policy, that is, the kind of individuals selected in the target island to be replaced by the immigrants:
    1. Worse individuals.
    2. Random individuals.
    3. Similar individuals.

  By similar individuals, one understands individuals with the same fitness as the immigrants.
- *MigrationInterval* represents the migration interval that is given from the percentage of reproduction cycles that need to be performed to initiate the exchange of individuals between islands.
- *Topology*: used to define communication between islands.
- *Synchronization*: defines whether the migration occurs synchronous or asynchronous.

There are two main ways to implement PIMs: homogeneous and heterogeneous. In the homogeneous models, all islands are configured identically. Homogeneity entails that all islands execute the same BA and have the same initial settings, resulting in all islands adhering to the same search strategy. On the other hand, in heterogeneous PIMs, islands are configured differently. Heterogeneity permits islands to use different settings and parameters, such as different population sizes and BAs. The idea is to explore different and possibly complementary parts of the search space solutions using diverse search strategies.

It is important to emphasize that the total populations in PIMs have a size equal to the sequential version since providing a more extensive search space in the parallel version would be unfair with the sequential ([45], [46], [24]).

In [45], the authors report that PIMs can provide better solutions than algorithms containing a single large population (Sequential version), both in performance and accuracy. The reason for the performance improvement is linked to *MigrationInterval* and *Topology*, respectively [41]. To improve accuracy, the authors show that the islands can explore different points in the search space of possible populations, something that the models containing a unique population cannot perform with mastery. In addition, a careful investigation into the impact of parameter setting needs to be made [39].

2.2.1 Communication Topologies

Topology is an essential factor in the performance of PIMs since it determines how slowly or quickly a solution disseminates to other islands. If the topology has dense connectivity, solutions will soon spread to all islands and can quickly take over the population. On the other hand, if the topology is sparse, the solutions will spread more slowly, allowing the emergence of different solutions; consequently, these solutions may subsequently form potentially better individuals. The Figures 2.6 (c), (d), and (e) represent examples of dense topologies, while Figures 2.6 (a) and (b) depict instances of sparse topologies. The ring topologies have been successful in various works (e.g., [47], [46], [48], [49], [13]). In addition, the topology also impacts the cost (run-time) of migration. Densely connected topologies can promote a better mix of individuals but also entail higher communication costs and have a high probability of reaching premature convergence.

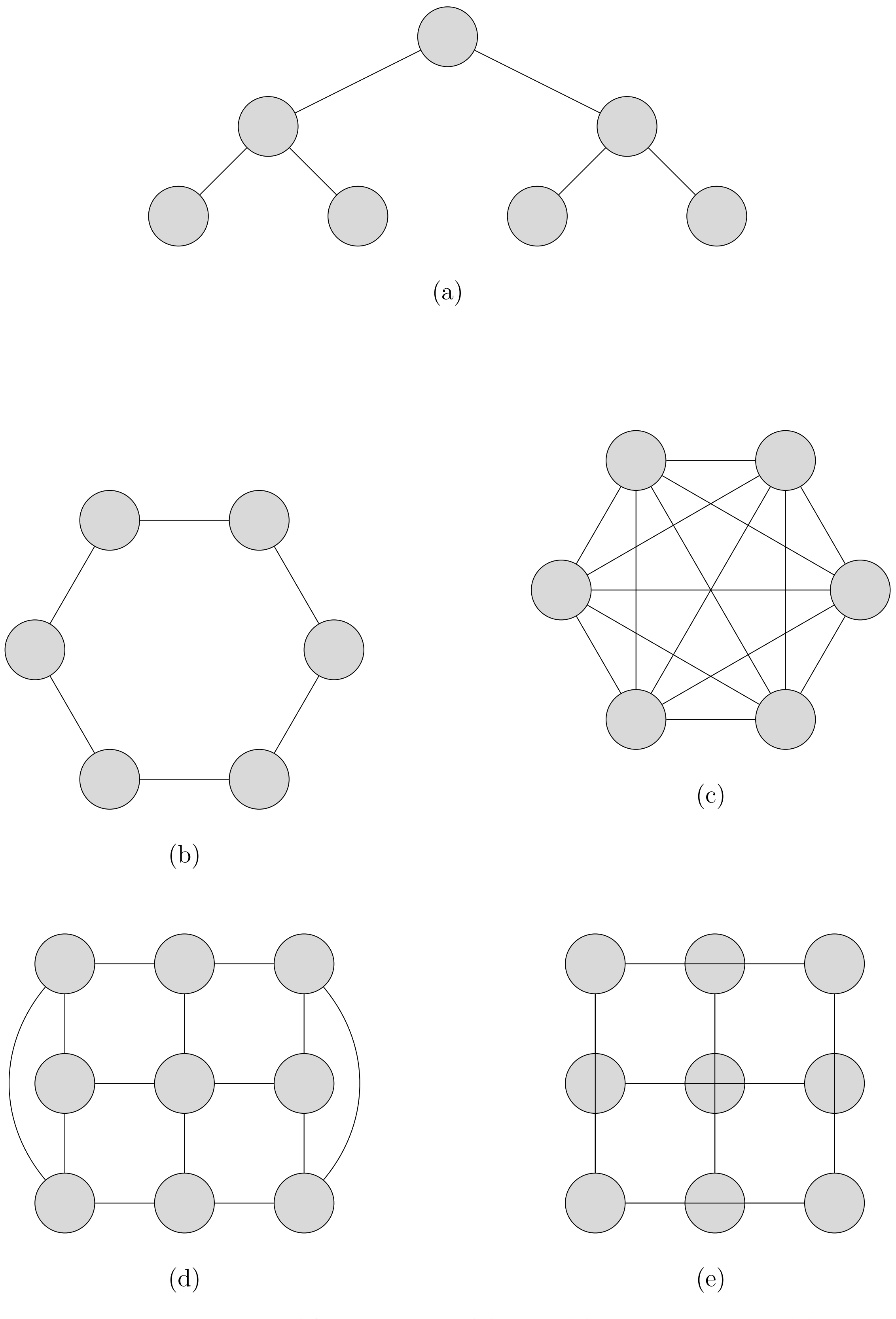


Figure 2.6: Static topologies: (a) binary tree, (b) ring, (c) complete graph, (d) $x \times y$-net where x and y represent rows and columns in the net, (e) torus.

Another crucial factor to consider when selecting a topology is whether the islands will be organized statically or dynamically. In static topologies, the arrangement is predetermined at the beginning of execution and remains constant throughout. On the other hand, dynamic topologies involve a more flexible approach, where communication between islands is not fixed but rather determined by specific criteria. The primary motivation for employing dynamic topologies is to pinpoint scenarios where migrating individuals are likely to have a discernible impact. The criteria for selecting an island as a destination can encompass factors such as population diversity measures [50], assessments of genotypic dissimilarity between islands [51], or even an attractiveness factor.

Proposed Dynamic Topology

Figure 2.9 depicts a dynamic complete graph topology we propose in [17]. This approach enables the creation of adaptive neighborhoods capable of adjusting based on the quality of solutions encoded in individuals and the diversity found within islands. In this methodology, all islands potentially have the ability to communicate with each other, making the topology resemble a complete graph. The difference is that neighborhoods change with each migration process, and not every island will communicate with all the others in the topology, so there are dashed edges in Figure 2.9. The quality and diversity of populations on each island were measured to apply dynamics. The islands are qualified as **good**, **bad**, and **medium**. The status good, bad, and medium are related to the diversity in the islands using two metrics: variance and average. The variance measures the diversity of each island, such that a high variance is associated with little resemblance between native individuals on the island. The average is related to the quality of the population on each island. Since the problems addressed are minimization, islands with low average fitness have the potential to be considered good islands. Then, for dynamic topologies, the algorithm adds these metrics for each island and ranks islands in decreasing order before migration starts.

Algorithm 1 Building dynamic links between islands

```
1: IdIsland, MediaAverage : array[1..$N_I$]
2: if Topology != Random then
3:     for $i = 1$ to $N_I$ do
4:         Average ← AveragePop($i$);            ▷ Average of population in island $i$
5:         Variance ← VariancePop($i$);          ▷ Variance of population in island $i$
6:         IdIsland[$i$] ← $i$;
7:         MediaAverage[$i$] ← Variance + Average;
8:     end for
9:     MergeSort(IdIsland, MediaAverage);    ▷ Indices in IdIsland change according to
       merge sort of MediaAverage
10:    if Topology == gbmm then
11:        for $i = 1$ to $N_I/2$ do
12:            Communication(IdIsland[$i$], IdIsland[$N_I - i + 1$]);
13:        end for
14:    end if
15:    if Topology == $\simeq$ then
16:        for $\boldsymbol{i = 1}$ to $\boldsymbol{N_I/2}$ do
17:            Communication(IdIsland[$\boldsymbol{2i - 1}$], IdIsland[$\boldsymbol{2i}$]);
18:        end for
19:    end if
20: else Random communication between islands;
21: end if
```

Algorithm 1 creates communications between islands in each migratory process according to the discussion in the previous paragraph. N_I is the number of islands, and the central idea is to create keys using the average and variance metrics (line 8) from the population of each isle. The *Mergesort* algorithm is used to sort the *IdIsland* arrangement according to the *MediaAverage* arrangement (line 9). It was adopted that the first, middle, and final third of the sorted islands in *IdIsland* are qualified as *good*, *medium* and *bad*, respectively (see Figure 2.7).

Before each migration, the communication between islands is generated as specified in Algorithm 1. Different neighborhood modeling strategies were implemented to diversify the dynamism in the proposed dynamic topology. In case the communication is $\simeq$, pairs of contiguous islands are communicated and, case it is gbmm, a pair of islands, which are selected from the beginning and end to the center of the sorted list of islands, are communicated (see Figure 2.8). While for *Random* communication, any island can communicate with any other island, regardless of rank.

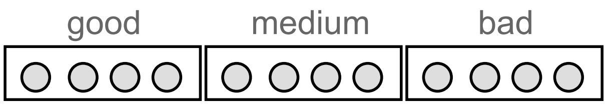


Figure 2.7: Classification of sorted islands.

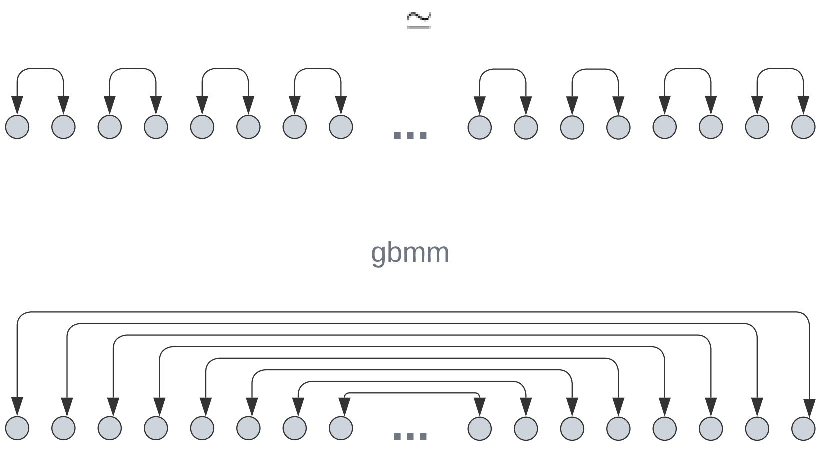


Figure 2.8: Organization of dynamic topology neighborhoods.

2.3 Case studies

In this section, the four case studies used for experiments are described.

2.3.1 Reversal Distance

A unichromosomal genome with n genes can be represented as a permutation $\pi = (\pi_1, \pi_2, ..., \pi_n)$, where $\pi(i) = \pi_i$, $1 \leq i \leq n$ and $\pi_i \in \{1, \ldots, n\}$. A reversal $\rho^{j,k}$, for $1 \leq j \leq k \leq n$ is an operation on π that inverts the sub string between π_j and π_k transforming π into $\pi' = (\cdots, \pi_{j-1}, \pi_k, \cdots, \pi_j, \pi_{k+1}, \cdots)$; for example: for $\pi = (1,4,3,5,8,2,7,6)$, $\rho^{5,7}(\pi)$ gives $\pi' = (1,4,3,5,7,2,8,6)$. There are two different types of permutations: signed and unsigned. In the unsigned case, genes are abstracted without any orientation, and in the

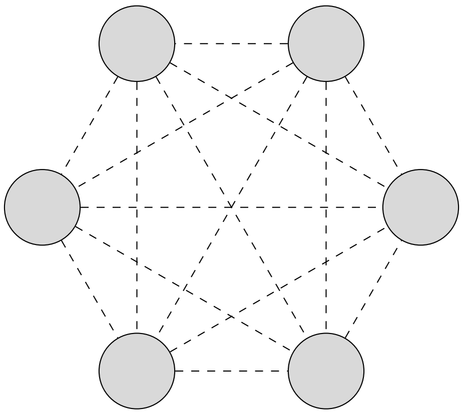


Figure 2.9: The dynamic topology resembles a complete graph, where all islands potentially have the capability to communicate. However, the choice of strategy defines an island's neighbors, which can be `gbmm`, $\simeq$, or `Random`.

signed case, each gene has a positive or negative sign reflecting its orientation. A reversal acts over a signed permutation by also inverting the genes' orientation in the reversed substring.

The *reversal distance* is the shortest length of a sequence of reversals that transform a permutation π into the identity permutation $\imath$, where $\imath$, is the permutation $(1, 2, \ldots, n)$, for the unsigned case, and $(+1, +2, \ldots, +n))$, for the signed case. The *sorting by reversals problem* determines the reversal distance of a permutation [52].

Example: *Consider the signed permutation.*

$$\pi = \{(-1, +6, +3, +4, -5, +2, +7)\}, \textit{ and}$$
$$\imath = \{(+1, +2, +3, +4, +5, +6, +7)\}$$

Observe that,

$$\begin{aligned}
\pi &= \{(-1, +6, +3, +4, -5, +2, +7)\} \\
\pi\rho_1 &= \{(\underline{+5, -4, -3, -6, +1}, +2, +7)\} \\
\pi\rho_1\rho_2 &= \{(+5, -4, -3, \underline{-2, -1, +6}, +7)\} \\
\pi\rho_1\rho_2\rho_3 &= \{(\underline{+1, +2, +3, +4, -5}, +6, +7)\} \\
\pi\rho_1\rho_2\rho_3\rho_4 &= \{(+1, +2, +3, +4, \underline{-5}, +6, +7)\}
\end{aligned}$$

Thus, after reversions ρ_1, ρ_2, ρ_3 and ρ_4 transform π into $\imath$ and reversion distance is 4.

The signed version of this problem, for short SRD, was proved to be in $\mathcal{P}$; namely, Bader, Moret, and Yan proposed a linear algorithm in [53]. Caprara proved that the unsigned version of this problem, for short URD, is $\mathcal{NP}$-hard [54]. The case study considered in this work is U . SRD has been extensively studied in the field of combinatorics of permutations (e.g., [55], [56]). Our algorithms apply Bader, Moret, and Yan's linear solution for SRD as the fitness function to solve URD, as done by Soncco, Muñoz, and Ayala-Rincón in [57].

2.3.2 Translocation Distance

An integer number represents a ge e. Signed integers model oriented genes and unsigned integers non-oriented gen s. A chromosome is a finite sequence of genes, and a genome is a set of chromosomes. Formally, a genome G with N chromosomes and n genes is a set $\{(x_{11}, \ldots, x_{1r_1}), \cdots, (x_{N1}, \ldots, x_{Nr_N})\}$, where $n = \sum_{k=1}^{N} r_k$ and $|x_{ij}| \neq |x_{kl}|$ whenever $i \neq k$ or $j \neq l$. For $1 \leq i \leq N$ and $1 \leq j_i \leq r_i$, if $x_{ij_i} \in [\pm n] = \{\pm 1, \ldots, \pm n\}$ then G is called a signed genome, whereas if $x_{ij_i} \in [n] = \{1, \ldots, n\}$ then G is an unsigned genome.

A translocation is an operation that acts over two chromosomes of a genome. There are two types of translocations: *prefix-prefix* and *prefix-suffix*.

Let $G = \{X_1, \ldots, X, \ldots, Y, \ldots, X_N\}$ be a genome and $X = (x_1, \ldots, x_l)$ and $Y = (y_1, \ldots, y_m)$ two chromosomes of G. A *prefix-prefix translocation* $\rho(X, Y, x_i, y_j)$, $1 \leq i < l$, $1 \leq j < m$, applied over G maintains the prefixes of X and Y and switches their suffixes, that is,

$$G\rho(X, Y, x_i, y_j) = \{X_1, \ldots, X', \ldots, Y', \ldots, X_N\}, \text{ where}$$

$$X' = (x_1, \ldots, x_i, y_{j+1}, \ldots, y_m) \text{ and}$$
$$Y' = (y_1, \ldots, y_j, x_{i+1}, \ldots, x_l).$$

Specifying whether G is a signed or an unsigned genome when one defines a prefix-prefix translocation is unnecessary once this concept is similar for both cases. A *prefix-suffix* *translocation* $\theta(X, Y, x_i, y_j)$, $1 \leq i < l$, $1 \leq j < m$, applied over G maintains the prefix of X and the suffix of Y and interchanges the suffix of X with the prefix of Y, that is,

$$G\theta(X, Y, x_i, y_j) = \{X_1, \ldots, X', \ldots, Y', \ldots, X_N\}, \text{ where}$$

$$X' = (x_1, \ldots, x_i, -y_j, \ldots, -y_1) \text{ and}$$
$$Y' = (-x_l, \ldots, -x_{i+1}, y_{j+1}, \ldots, y_m),$$

when G is a signed genome and

$$X' = (x_1, \ldots, x_i, y_j, \ldots, y_1) \text{ and}$$
$$Y' = (x_l, \ldots, x_{i+1}, y_{j+1}, \ldots, y_m),$$

when G is an unsigned genome (See Figure 2.10).

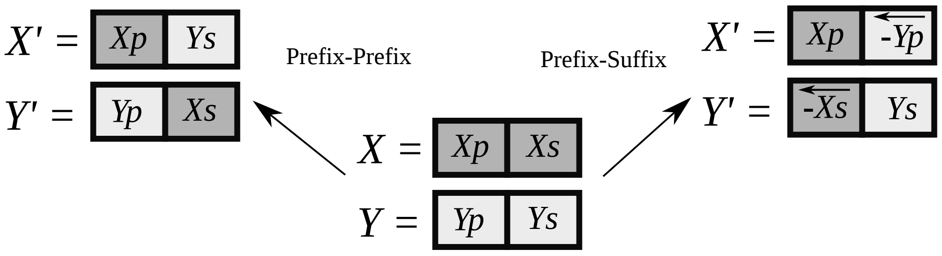


Figure 2.10: Prefix-prefix and prefix-suffix translocations over a signed genome.

Given a chromosome $X = (x_1, \ldots, x_l)$ the elements of the set $\{x_1, -x_l\}$ are called *tails* of X, if X is a signed chromosome, whereas the elements of the set $\{x_1, x_l\}$ are the tails of X in the unsigned case. Two genomes are considered co-tails if they share the same set of tails. It's important to note that if G is a genome and ρ is a translocation, then $G\rho$ and G are co-tails. Therefore, to guarantee that a genome A can be transformed into a genome B by translocations, it is necessary to ensure that A and B are co-tai s. The two variations of the genome rearrangement problem through translocations are defined as follows.

- **Unsigned Translocation Distance Problem (UTD):** given two unsigned co-tails genomes A and B with the same number and over the same set of genes, determine the minimum number of translocations needed to transform A into . One can assume that the genes of B are in increasing order. One calls B an identity genome.

- **Signed Translocation Distance Problem (STD):** it is defined analogously to UTD. Still, in this case, A and B are signed co-tails genomes, and the genes of the signed identity genome B are positive and in increasing order.

If $\rho_1, \ldots, \rho_k$ is a shortest sequence of translocations such that $A\rho_1 \ldots \rho_k = B$ then k is called the *translocation distance* between A and B. Since the identity is sorted, the UTD and STD problems from A to an identity (that is, co-tail with A) are also referred to as the problem of *sorting* genome A by translocations.

Example: Consider the signed genomes

$$\begin{aligned} A &= \{(+1,+3),(+4,-8,+5,+2,+6),(+7,+9)\}, \text{ and} \\ B &= \{(+1,+2,+3),(+4,+5,+6),(+7,+8,+9)\} \end{aligned}$$

Observe that,

$$\begin{aligned} A \quad &= \{(+1,\underline{+3}),(+4,-8,+5,\underline{+2,+6}),(+7,+9)\} \\ A\rho_1 \quad &= \{(+1,+2,\underline{+6}),(+4,-8,+5,\underline{+3}),(+7,+9)\} \\ A\rho_1\rho_2 \quad &= \{(+1,+2,+3),(+4,-8,\underline{+5,+6}),(\underline{+7},+9)\} \\ A\rho_1\rho_2\theta_1 \quad &= \{(+1,+2,+3),(+4,\underline{-8,-7}),(\underline{-6,-5},+9)\} \\ A\rho_1\rho_2\theta_1\theta_2 &= \{(+1,+2,+3),(+4,+5,+6),(+7,+8,+9)\} \end{aligned}$$

Thus, the prefix-prefix translocations ρ_1 and ρ_2 together with the prefix-suffix translocations θ_1 and θ_2 transform A into B and translocation distance between A and B is 4.

UTD is addressed in this work and was shown to be $\mathcal{NP}$-hard by Zhu and Wang in [58]. As for SRD, the Signed Translocation Distance problem (STD) considers signed genomes and belongs to $\mathcal{P}$. Indeed, algorithms of time complexity $O(n^3)$ and $O(n)$ were respectively designed by Hannenhalli [59] and by Bergeron, Mixtacki, and Sotye in [60]. In the proposed approaches, the fitness function to solve the UTD problem is computed using the algorithm proposed in [60].

2.3.3 N-Queens

This problem aims to place n queen chess pieces in an $n \times n$ chessboard in a configuration where no two queens can attack each other. The problem is related to the N-Queens completion problem that consists in completing a configuration from a given partial solution and is known to be both $\mathcal{NP}$-complete and $\#\mathcal{P}$-complete (see [61]). A relevant advantage of this problem to test optimization algorithms is that it is well-known each $n \times n$ chessboard, for $n > 3$, has solutions with zero attacks.

This problem is commonly used as a benchmark in different machine learning-based methods. For example, it has been used in constructive backtracking algorithms ([62]) and BAs-based meta-heuristics such as GA ([63]) and PSO ([64]).

The search for a solution corresponds to a minimization problem. Considering the search space is given by n different positions in an $n \times n$-chessboard, its size is given by $(n^2)!/(n!\,(n^2-n)!)$ and the measure of each possible solution is defined as the number of attacking movements between queens at these positions. Here, each candidate placement solution is encoded as an n-dimensional vector $\boldsymbol{x}$, where the pair (j, x_j), for j an index of $\boldsymbol{x}$, represents the placement of a queen onto the jth row and x_jth column of the chessboard. Notice that, in this manner, the search space is reduced to a set S of n^n possible placements. The goal is to minimize a function $f_{attack} : \mathrm{S} \to \mathbb{N}$ that maps each $\boldsymbol{x} \in \mathrm{S}$ into the number of pairs of queen pieces under mutual attack given by the placement $\boldsymbol{x}$. The function f_{attack}, used as a fitness function, is linearly computed. Figure 2.11 (taken from [20]) illustrates the placement of a single queen piece at (4, 4) position on an 8×8 board and Figure 2.12 shows a placement where none of the 8 queen pieces attack each other.

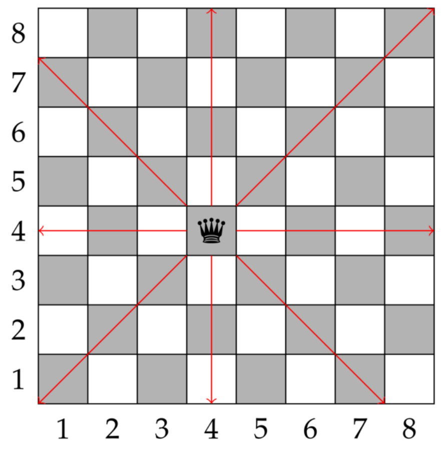


Figure 2.11: Movement of a queen chess piece on an 8×8 board.

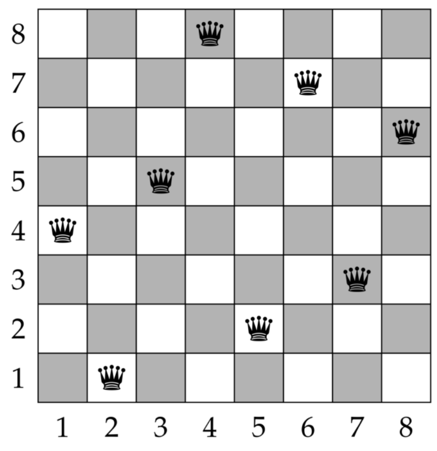


Figure 2.12: A placement $\boldsymbol{x}$ where $f_{attack}(x) = 0$.

2.3.4 Task Mapping and Scheduling

The *Task Mapping Problem* (TMP) consists of mapping tasks of a *Real-Time Application* (RTA) onto one of the multiple processor cores of a *Real-Time System* (RTS). The correctness of an RTS depends on its compliance with deadlines.

A type of multiple processor, *Multiprocessor System-on-Chip* (MPSoC) ([65]) is considered in this case stu y. MPSoCs have a communication architecture comprising a *Network-on-a-Chip*, a well-established intra-chip communication architecture paradigm [66]. Due to the real-time setting, the *Network-on-a-Chip* fits the category of a *Real-Time Network-on-a-Chip* (RTNoC) as defined by [67]. The RTNoC consists of a wormhole packet switching strategy and virtual channels, allowing the preemption of data flows during transmission by ones with a higher priority level.

For static schedulability analysis of feasible mapping solutions, this case study uses a specific model for the RTNoC-based MPSoC platform and the RTA mapped onto it, similar to the one by [68]. The TMP is $\mathcal{NP}$-hard similar to the knapsack problem given by [69]; thus, a brute force approach is unfeasible. Indeed, a case where n tasks are being mapped onto a system with m processors gives a search space with m^n possible task mapping solutions.

The platform model, $\Psi = \langle \Pi, R, \Lambda \rangle$, consists of $\Pi = \{\pi_1, ..., \pi_n\}$, a set of homogeneous processing elements connected to its network interface elements, and a set of routing elements, $R = \{\rho_1, \rho_2, ..., \rho_n\}$, to switch messages that connect elements in Π and themselves using $\Lambda = \{\lambda_{\pi_1,\rho_1}, \lambda_{\rho_1,\pi_1}\lambda_{\rho_1,\rho_2}, ..., \lambda_{\rho_n,\rho_{n-1}}\}$, a set of unidirectional links in a regular $i \times j$ processors mesh-grid topology, where $ij = n$. The model comprises a set of tasks Γ, such that $\tau \in \Gamma$ is given as a tuple $\langle C, T, D, P, \phi \rangle$ formed by the worst-case running time, inter-arrival period, relative deadline time, priority-level, and a transmitted message ϕ, respectively. At the end of its execution, the task τ may transmit a message ϕ towards another task. The schedulability analysis considers the utilization factor of processor cores and links. The utilization factor for a processor core, U_π, presents the ratio in which the processor is used by the tasks mapped onto it as in [70].

The same principle is used to evaluate the utilization rate of links in the RTNoC to check whether they do not exceed their maximum communication bandwidth.

This simple analysis is necessary to avoid non-schedulable solutions demanding resources exceeding the available capacity, implying that tasks and messages do not comply with deadlines. This situation holds for any given processor π if its utilization factor is such that $U_\pi > 1$ and for any given link λ such that $U_\lambda > 1$. However, the utilization analysis is insufficient. Even in cases where all processors and links are not over-utilized, the task mapping may not be fully schedulable depending on the task set in the mapped RTA. Even though these tests are insufficient, they are helpful for quickly identifying whether mappings suit the time requirements.

Given a set of tasks Γ and a platform Ψ, a task mapping solution can be encoded as an n-dimensional vector of positive integers $\boldsymbol{x}$, where each component, x_j, represents the index of the processor in the platform Ψ responsible for processing the task $\tau_j \in \Gamma$. The search-based optimization for a task mapping solution that reduces the number of overused resources is expressed by a function f_{util} that calculates the number of processors and links overburdened in the system given a task placement $\boldsymbol{x}$. The minimization goal using f_{util} as a fitness function is to find a task placement $\boldsymbol{x}^*$ that respects processors and links maximum capacities.

Chapter 3

Overview of Preliminary Research

This chapter provides a comprehensive overview of our preliminary research. In Section 3.1, we begin the discussion regarding HoPIMs, examining the sensitivity of migration parameters and the methods for creating initial populations for the islands. Section 3.2 explores the evolution of HoPIMs, which have become integral components of recognized BAs in the literature, serving as evolutionary engines within islands for quality and performance comparisons. Related studies on HoPIMs are illustrated in Section 3.3. Section 3.4 ventures into HePIMs, initially discussing the conventional island method that allocates various evolutionary engines. Following that, Section 3.5 delves into an approach focused on reconfigurable HePIMs. We conclude the chapter by presenting a mind map that organizes and elucidates the published works in the research involving both homogeneous and heterogeneous models, as detailed in Section 3.6.

3.1 Evolution and Calibration of Homogeneous Parallel Island Models (HoPIMs)

Our research involving PIMs began in [12], where approaches to address the UTD problem were proposed. One of these approaches utilized versions of HoPIMs with an evolutionary engine based on a memetic algorithm developed by the renowned `GA` for tackling the UTD problem [23]. However, the solutions obtained with the model proved to be unsatisfactory. The HoPIM used a topology based on a complete graph (see Figure 2.6 c), where the best individual was shared with all other islands in each generation to replace the worst native individual. It proved to be inadequate as it contributed to premature convergence. Another critical point to highlight is that the unsatisfactory performance was related to using the same breeding cycle parameter values used by the sequential algorithm. The experience gained throughout the work allowed us to understand that HoPIMs require special care both in defining migration policies and in selecting the parameters of the algorithms responsible for the evolution of populations.

Our article [13] focused on exploring the parameters involved in island models to optimize solutions. The GA continued to serve as the evolutionary engine for the proposed HoPIMs, notable for being a widely recognized algorithm in the literature for addressing the URD problem [71]. The initial phase involved calibrating migration and evolution parameters using a greedy method. This approach primarily focuses on identifying suitable values for reproduction and migration before advancing to the calibration stage. Once determined, these values are utilized and remain constant across subsequent generations. The so-called "taxonomy T1" was adopted for calibration as discussed in [72]. The tuning method employs an iterative approach. The algorithm's parameters are organized into a vector, and iteration occurs by traversing each vector position, representing a specific parameter. In each iteration, various possible values are tested for the current parameter, and the value that results in a better solution quality is fixed. The procedure is repeated for each parameter, with the peculiarity that once a compelling value for a parameter is found, this value is used as a reference for searching one optimized value for the following parameter. Thus, the method unfolds, adjusting each parameter until all are tuned. In this work, HoPIMs employing ring and complete graph topologies (refer to Figures 2.6 b and 2.6 c) for the URD case study are proposed. The findings demonstrate that even with a greedy method, a systematic approach to parameter tuning significantly increases the likelihood of discovering superior solutions compared to the sequential version.

Ultimately, the greedy method provided parameter values that were better suited for the case study resolution. An important aspect was to assess the behavior concerning initially randomly generated populations and those drawn from the sequential version, where the population from the sequential `GA` is divided into equal-sized partitions and distributed among the islands. The experiments demonstrated that through straightforward calibration, as the method employed is greedy, it is possible to achieve superior solutions compared to the sequential version. Furthermore, it became evident that the HoPIM employing a ring topology provides better solutions than the HoPIM using the complete graph topology. Such a behavior could be explained by the former topology being sparser than the latter. Such experiments also determined a milestone in our research, leading us to adopt the sequential population to initialize populations on the islands, given that the results were, on average, superior to those obtained by generating random populations for each island.

The speedup was satisfactory. The ring topology yielded the best speedup results, reaching around 11, as expected due to its sparser nature. In contrast, the complete graph topology, being denser, still provided a good speedup, above 7. Models employing partitioned populations introduce an overhead that negatively impacts runtime, with a lesser effect on the ring topology model and a more significant degradation in the complete graph topology model.

3.2 HoPIMs: Synchronous/Asynchronous, Migration Strategies, Parameter Evaluation, BAs

In [17], [18], [20], a variety of HoPIMs were evaluated using `GA` as the evolutionary engine. Synchronous and asynchronous HoPIMs were proposed to solve the four cases of study: UTD, URD, N-Queens, and TMP. Some discrepancies found in the literature regarding HoPIMs were addressed, primarily concerning the behavior of synchronous versus asynchronous versions, the number of islands, and the impact of parameters on accuracy and speedup.

Initially, we calibrated the parameters of each HoPIM separately using a greedy strategy, as done in [13], to ensure reliable specific configurations. Such a calibration allowed us to conduct a fair and robust analysis of the overall behavior of each HoPIM, whether synchronous or asynchronous. Synchronous and asynchronous HoPIMs were implemented based on the topologies in Figures 2.6 and 2.9 with 12 and 24 islands. All these models underwent parameter calibration and consistently provided higher-quality solutions than the sequential version. The experiment results did not identify a generic HoPIM that offered the best speedup and accuracy for all case studies. Regarding accuracy, synchronous HoPIM delivered the best solutions in most instances. Synchronous static 12-island HoPIMs proved the most competitive in solving URD, while static and dynamic 12-island HoPIMs excelled in UTD resolution. For TMP, the most effective solutions came from synchronous dynamic 24-island HoPIMs, and in the case of N-Queens, the best results were obtained through a synchronous static binary tree 24-island HoPIM.

The experiments over the four case study problems also confirmed various issues discussed in the literature. Our primary goal was to determine whether a uniformly executing island model with the same BA, the same number of islands, and the same topology would yield good adaptation in terms of accuracy and speedup for all problems. The main lesson learned is that the best-adapted HoPIM depends entirely on the specific problem. An important observation is that variations in the number of chromosomes in the genome can significantly impact the choice of the most effective HoPIM.

Although our primary focus has been on accuracy, it is noteworthy that all HoPIMs provided significant speed gains. In some cases, these gains were so efficient that they achieved linear speedup, indicating that the performance improvement was proportional to the increase in the number of processors. This outcome underscores the effectiveness of PIMs in the addressed case studies.

The existing literature frequently indicates that asynchronous HoPIMs yield superior speedup compared to their synchronous counterparts (e.g., [73, 74, 11]). However, the experiments revealed that this statement is only valid when all models share the same parameter configuration. The parameters obtained during the calibration phase, focused on improving accuracy, also influenced the overall speedup of the HoPIMs. For example, modifications on the *selection* and *crossover* parameters showed a significant impact. Furthermore, migration parameters, such as *MigrationInterval* and *NumMigIndividuals*, were identified as the most sensitive, as they control the overhead in the HoPIMs. Finally, evaluating migration topologies highlighted the potential of dynamic topologies when focused on compelling exploration. Nevertheless, there was no consensus on the best solutions for all problems, and determining a single superior topology, whether static or dynamic, was impossible. The only clear conclusion is that sparser topologies, such as ring, tree, and $x \times y$-net, outperformed denser topologies, like complete graphs and torus.

Based on the feedback received and the information in the literature, our research continues exploring the impact of solution quality when altering the evolutionary mechanisms within the HoPIMs [19]. The new experiments included three evolutionary engines: `SSA`, `PSO` and `GA` and the topologies shown in Figures 2.6 and 2.9, applied to the URD case study. The HoPIMs derived from `SSA` and `PSO` underwent a parameter calibration phase, similar to the process used for `GA`-based HoPIMs. HoPIMs generated from `SSA` resulted in low-quality solutions; furthermore, when considering 24 islands, a higher degradation was observed than in models with 12 islands. Concerning the quality of HoPIMs generated using `PSO`, static topologies stood out, particularly for small to medium-sized URD instances, surpassing `GA`-based HoPIMs. However, these HoPIMs were less competitive in dealing with larger URD instances.

3.3 Related Work in Homogeneous Island Models

This section will primarily delve into works about HoPIMs. We have chosen to omit discussions related to HePIMs at this point, as those will be thoroughly examined in the upcoming chapter.

Saito *et al.* [75] introduced an approach to parallelize decomposition-based multi-objective evolutionary algorithms (MOEA/D). The technique utilizes many-core environments, such as Graphics Processing Units (GPUs), to prevent degradation in solution accuracy. Degradation is mitigated by defining a virtual overlapping zone between partitions and the selection of individuals. Weight vectors of adjacent partitions are used to evaluate individuals in this zone for mating and migration. The method is compared with sequential MOEA/D, parallel MOEA/D without migration, and parallel MOEA/D with migration. The case study focused on a two-objective knapsack problem with constraints. The results highlighted the effectiveness of the proposed method in enhancing solution diversity and speedup. It is important to note that our work does not explore multi-objective problems. However, this study emphasizes the potential of island models to address such problem domains.

Another research aiming at preventing failures in refrigerated display cases installed in convenience stores and supermarkets was proposed by Otaka *et al.* in [76]. The study introduces a fault detection method using folder-based artificial neural networks, employing a modified brainstorm optimization with PIMs and correntropy (see [77]). Variations with static ring topology, trigonal pyramid, and cube topologies were explored. They implemented an elitist migration policy where the best immigrants replace the worst natives every ten generations. Experiments were conducted with limited numbers of islands: two, four, and eight. Statistical tests highlighted the superiority of the four-island trigonal pyramid topology model, indicating a success rate of approximately 99.4% for the presented samples. This work shares similarities with the approach we are adopting in our research, including variations in topologies and the number of islands. The difference is that the authors opted for parameter values without undergoing a parameter calibration, which, according to our study, significantly affects the quality of solutions.

Ohira *et al.* [78] proposed a HoPIM using a GA as an evolutionary mechanism for GPU deployment. The approach clusters similar individuals on each island to focus on a specific search space region. The similarity was calculated using the longest common subsequence distance ([79]), in contrast to genotype similarity in our approach, justified because it is more suitable for the type of problems we address. Unlike a parameter-guided migration, in the proposed approach, at each migration, the global population is grouped into subpopulations and subsequently distributed among islands based on their similarity. The approach establishes a dynamic communication topology based on similarity, while the migration interval is influenced by diversity. The implementation occurs on GPU and CPU, demonstrating acceleration improvements as the number of islands increases, with particularly significant acceleration in the CPU version. The outcomes obtained by the authors reinforce our observations regarding accuracy and speedup, underscoring that speedup has the potential for scalability exclusively. At the same time, an identical situation does not occur for accuracy.

Skakovski and Jędrzejowicz propose an island model composed of islands of different sizes that operate independently without any communication among themselves [80]. The model successfully executes the DE algorithm on its islands to solve the discrete-continuous scheduling problem. Our work has not explored such a strategy, as we have chosen not to work with islands of different sizes. Subsequently, the authors in [81] introduced dynamism by allowing adjustments to the island population during the evolutionary cycle and a migration policy. Abed-Alguni *et al.* developed a series of works involving homogeneous HoPIMs from the Cuckoo Search (CS), Grey Wolf Optimizer (GWO) and Whale Optimization Algorithm (WOA), respectively. In [82], they proposed a HoPIM from the GWO [34]. The model allows the exchange of individuals between islands based on a random ring topology that is altered at each migration process. Such a dynamic topology does not have a defined criterion for organization, unlike the one applied in our proposed dynamic topology in [17]. The adopted migration policy is grounded in an elitist and compared to swarm-based optimization algorithms: its sequential version, Cuckoo Search [33], Memory-Based Hybrid Dragonfly [83], Fireworks Algorithm with Differential Mutation [84], and Adaptive Differential Evolution with Linear Population Size Reduction Evolution [85]. The results indicate that the HoPIM produces competitive outcomes compared to the other evaluated algorithms. However, it did not compute superior solutions to the ones computed by the competitors for all assessed problems. Additionally, the sensitivity of the island model to its parameters was evaluated in various scenarios, including migration frequency (5% and 10%) and the number of islands (two, five, and ten). The results suggest that the HoPIM with ten islands and a migration frequency of 5% delivers the best outcomes. Later, the HoPIM was adapted to deal with discrete problems [86]. In [87], Abed-Alguni *et al.* investigated an alternative evolutionary approach within the context of the HoPIM presented in [82]. The model was adjusted to evolving islands using the WOA. A comparative analysis involving other HoPIMs suggests that the model proposed by the authors enhances result accuracy. Moreover, sensitivity analysis of its parameters reveals that the convergence behavior is intricately linked to the migration parameters. It is important to note that the authors did not conduct a comparative study with the HoPIM proposed in [82], which uses GWO, a potential avenue for further investigation into the behavior of both models. Recently, Abed-Alguni *et al.* [88] employed the CS as the evolutionary engine for islands in the HoPIM presented in [82]. The model's performance was deemed efficient in tests conducted in continuous and discrete contexts. The experiments revealed that this approach yielded superior solutions to the model utilizing the GWO proposed in [82].

Thaher *et al.* propose, in their work [89], a HoPIM based on a Crow Search algorithm. The model employs a dynamic communication topology organized in a ring, which is altered with each migration process. The authors work with the population percentage in migration, and the migration frequency is controlled by the number of generations, following an elitist policy where the best immigrants replace the worst natives. Experiments were conducted using well-known benchmarks in the literature, containing optimization functions with optimal solutions found in polynomial time. For comparison, the model was subjected to experiments with various BAs. The experiment scenarios found in [82, 87, 88], where variations in the number of islands and migration frequency are used to observe the behavior of the models. The authors' observations emphasize that migration parameters significantly influence the results; thus, parameters must be carefully chosen to offer high-quality solutions. Furthermore, the comparison with other BAs demonstrates that the HoPIM can provide competitive results and achieve the best outcomes in most cases.

3.4 Comparative Analysis of Homogeneous and Heterogeneous Models

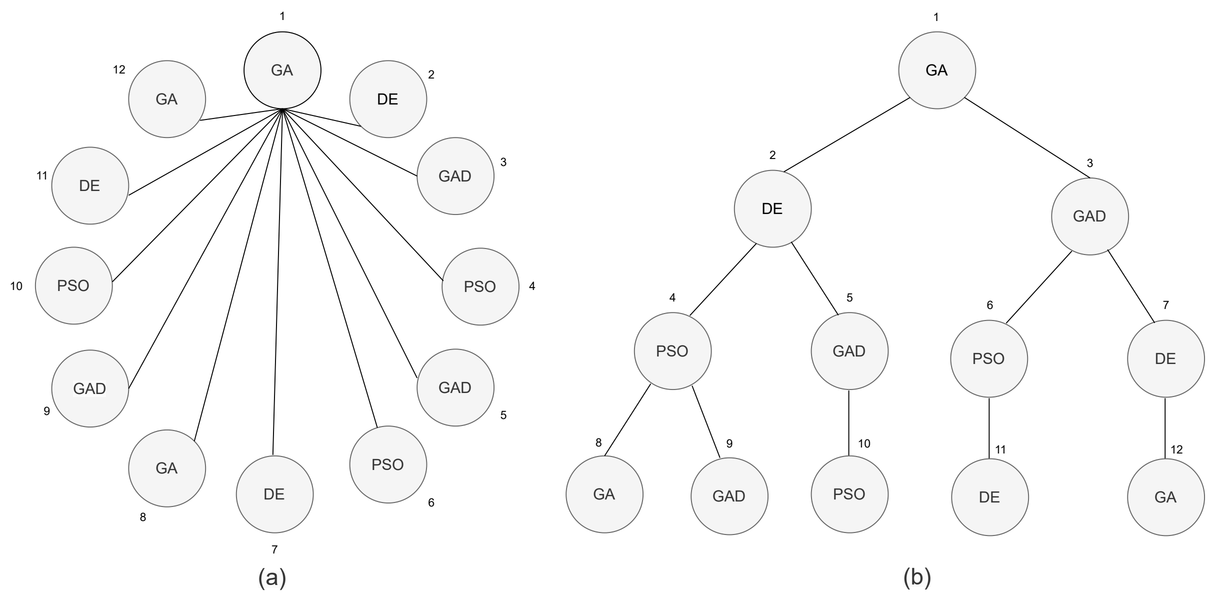


Figure 3.1: (a) HePIM organized with a complete graph topology. For simplicity, only a subset of edges. (b) HePIM configuration using a binary tree topology.

The HePIMs have been a focal point of discussion in our subsequent research reported in [14]. The $\mathcal{NP}$-hard problems are notable for having a vast and challenging search space. Adopting HePIMs, which can integrate various BAs, underscores a promising potential to address these challenges. The intrinsic diversity of these models provides a range of exploration methodologies within the solution space, enabling adaptation to different characteristics and nuances of the problem. This flexibility makes HePIMs a valuable approach in pursuing practical solutions amid the inherent complexity of $\mathcal{NP}$-hard problems.

In our initial research with HePIMs, we employed `PSO`, `GA`, `GAD` and `DE` as evolutionary engines to tackle instances of the URD problem. We adopted synchronous and asynchronous 12-island HePIMs using a communication topology based on a static binary tree (Figure 2.6(a)) and another based on a dynamic complete graph (Figure 2.9). Each BA was assigned to three islands (see Figure 3.1), and an equal number of generations was maintained for both homogeneous and heterogeneous PIMs. Experiments with HoPIMs were required to evaluate the quality of proposed HePIMs.

Let $\mathcal{P}^{\text{Het}}_{\text{Tr12S}}$, $\mathcal{P}^{\text{Het}}_{\text{gbmm12S}}$, $\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$ denote the synchronous and synchronous HePIMs created from the topologies of binary tree and dynamic complete graph, respectively. The subscript prefixes `Tr` and `gbmm` denote that the model uses the static binary tree and dynamic complete graph topology, respectively, and the subscript suffixes `12S` and `12A` denote the number of islands and whether the model is synchronous or asynchronous. The experiments indicate that synchronous and asynchronous HePIMs deliver better solutions than homogeneous models when considering `PSO`, `GA` and `GAD`. The only exception is HoPIMs based on `DE`. Furthermore, asynchronous HePIMs demonstrated superior solutions to synchronous HePIMs (see details in [14]). The two figures 3.2 and 3.3, and the two figures 3.4 and 3.5 present a comparative analysis of results between HoPIMs and HePIMs. The first two figures are for synchronous, and the latter two figures are for asynchronous PIMs. HoPIMs are distinguished by their names as superscripts. At the same time, other details are explained similarly for HePIMs, including topology, number of islands, and synchronization. Instances of the URD problem with sizes 100, 110, . . . , 140, and 150 were employed to evaluate these models. It's worth noting that URD is a minimization problem, so the closer to the center, the better the quality of results. Notably, HePIMs are surpassed only by homogeneous models implemented from `DE`, and homogeneous and heterogeneous versions, utilizing dynamic topology, deliver superior solutions than versions based on static topology.

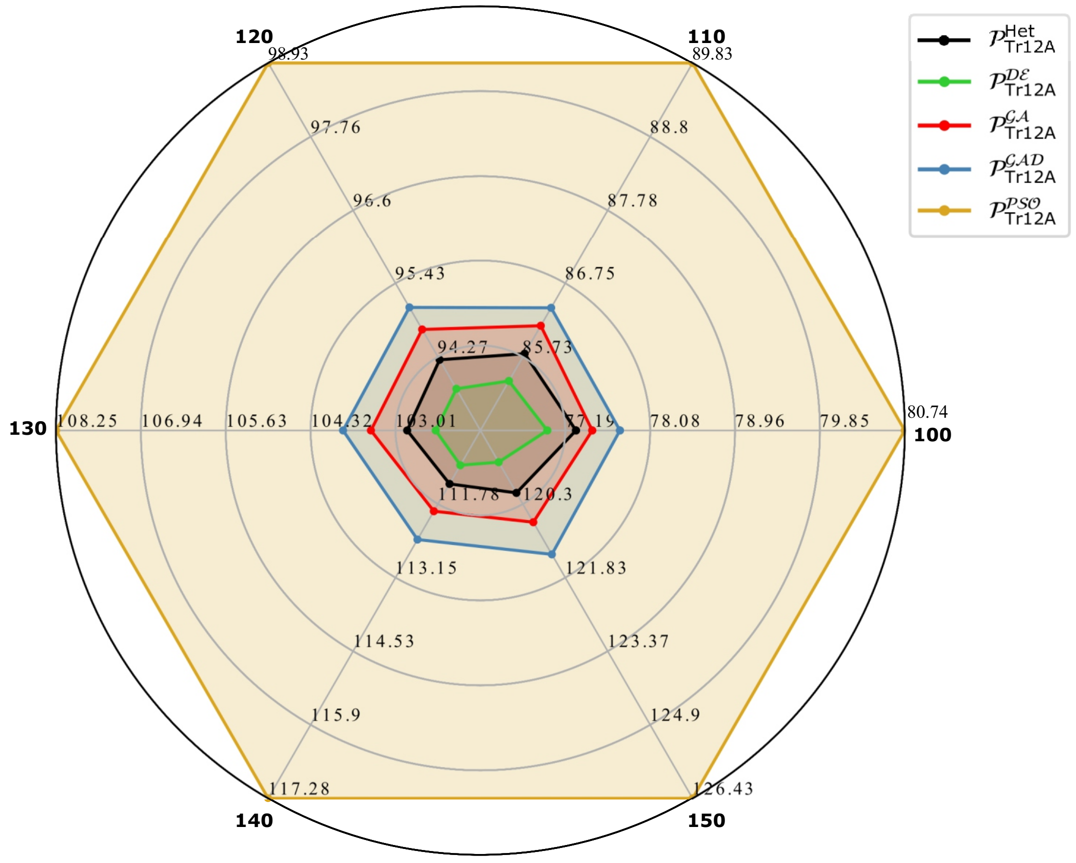


Figure 3.2: Accuracy of $\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$ and related HoPIMs

3.5 Reconfigurable Heterogeneous Models

In [15], we present reconfigurable heterogeneous PIMs, building upon the work's continuity in [14]. We focused exclusively on asynchronous HePIMs, which have proven to provide more effective solutions for the URD problem than synchronous HePIMs. This reconfiguration concept involves replacing the BA of the island with the worst accuracy with the BA of the island that presents the best accuracy at predefined intervals determined by the number of generations. These intervals are identified through calibration experiments conducted during a setup phase. A master-slave model is utilized to classify islands based on their fitness average and variance, employing this information to construct neighborhoods throughout the evolutionary process. The master island (Island 1) receives data from all islands; it ranks and notifies the worst island. The worst island reconfigures (i.e., replaces) its BA with the BA of the best-ranked island. The reconfiguration process is

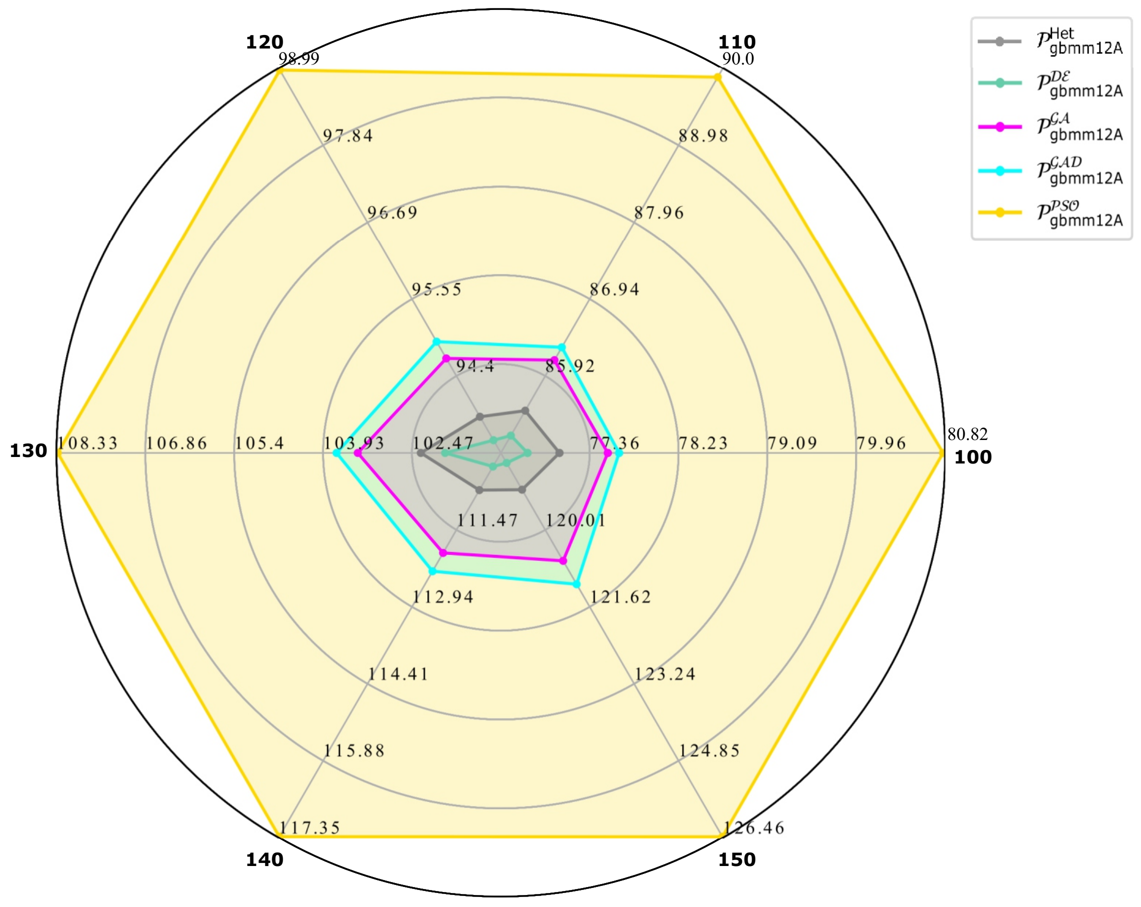


Figure 3.3: Accuracy of $\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$ and related HoPIMs.

controlled by a parameter called reconfiguration frequency *(RF)*. The *(RF)* dictates the number of evolutionary generations for periodic reconfiguration. It is established through a parameter calibration phase preceding the experiments. We implemented two HePIMs: $\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$. Both models utilize the reconfiguration methodology, with the former adopting a static binary tree topology (Figure 2.6 (a)), while the latter employs a dynamic complete graph topology (Figure 2.9). Operating asynchronously, they evolve through a refined migration policy, enabling individuals to exchange, thereby preserving diversity. Figure 3.6 illustrates reconfiguration cycles, using the $\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$ model as an example.

Reconfigurable HePIMs, especially the $\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$ consistently demonstrated superior solutions compared to the traditional method of implementing HePIMs used in [14]. However,

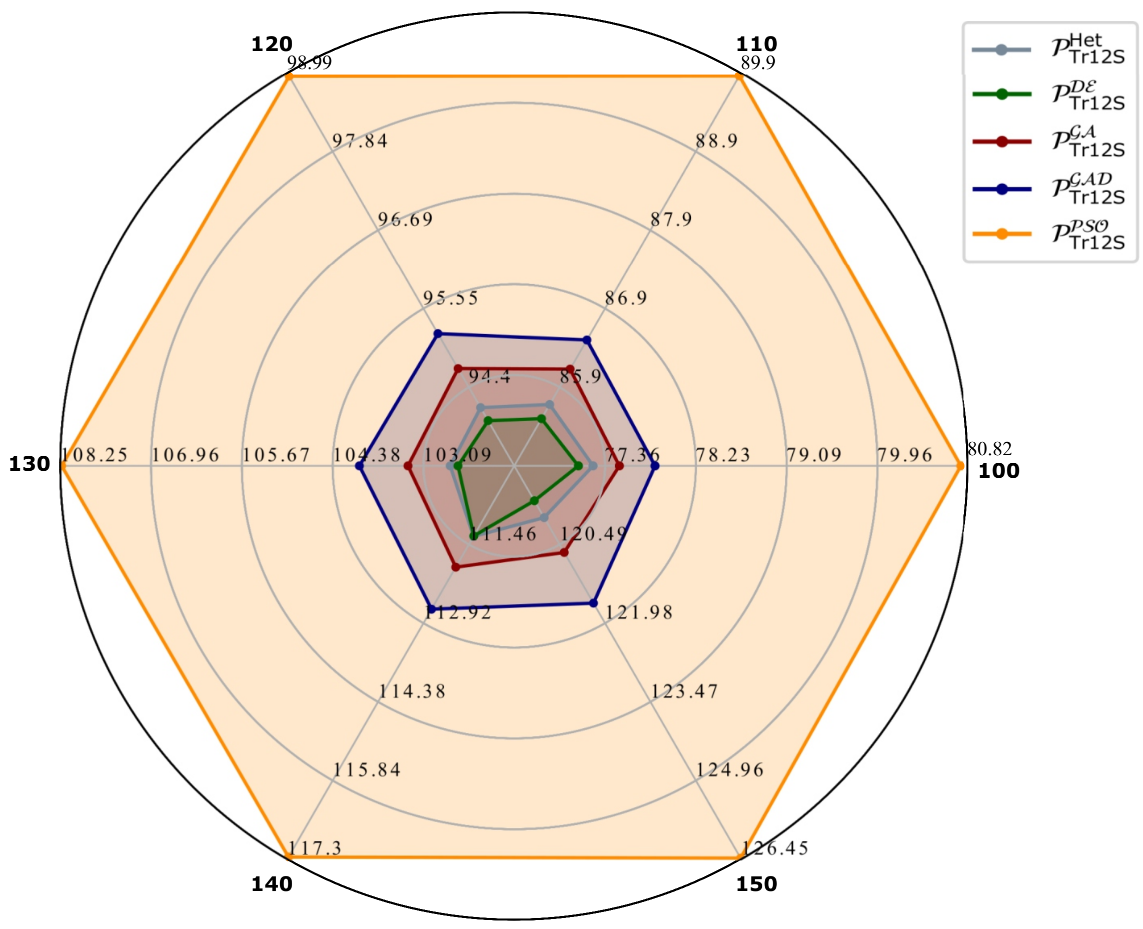


Figure 3.4: Accuracy of $\mathcal{P}^{\text{Het}}_{\text{Tr12S}}$ and related HoPIMs.

the HoPIM utilizing `DE` stood out by exhibiting superior efficiency, delivering improved solutions across most input datasets (see Figures 3.7, and 3.8). A more detailed analysis of the results reveals that the solution quality of HoPIMs based on `DE` consistently surpasses that of other HoPIMs, especially when compared to those derived from `PSO`. Such results posed a significant challenge for HePIMs to overcome, even with various calibrations in migration parameters.

Including an evaluation of speedup performance, the reconfigurable HePIMs maintain competitiveness compared to traditional HePIMs, as evidenced in Table 3.1. HoPIMs' speedups are calculated according to their sequential versions, while speedups for non-reconfigurable and reconfigurable HePIMs are based on the average runtime of sequential BAs: `PSO`, `GA`, `GAD`, and `DE`. The additional cost associated with reconfiguration can be

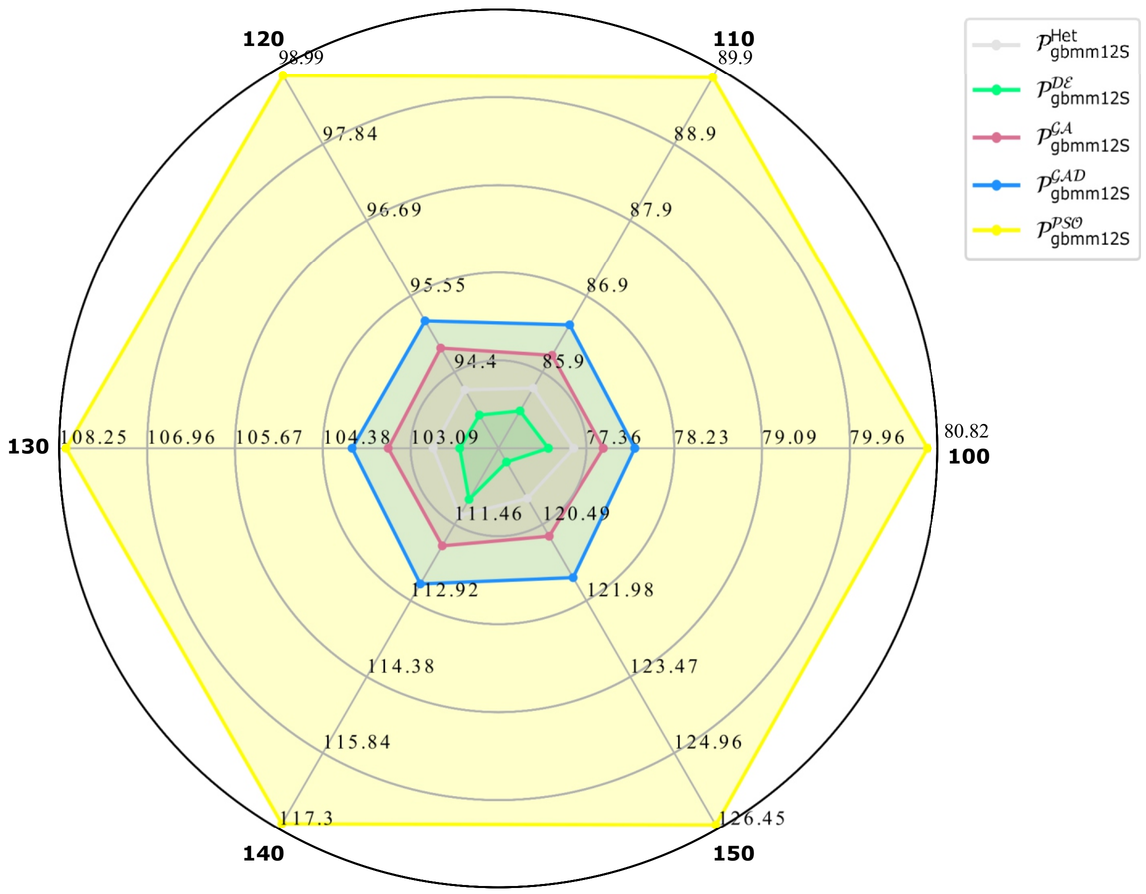


Figure 3.5: Accuracy of $\mathcal{P}^{\text{Het}}_{\text{gbmm12S}}$ and related HoPIMs.

quantified by the speedup ratio between reconfigurable and non-reconfigurable HePIMs: 0.87 for $\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$ and 0.88 for $\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$.

Table 3.1: Speedups for reconfigurable HePIMs.

Homogeneous								Heterogeneous			
GA		GAD		PSO		DE					
Tr12A	gbmm12A	Tr12A	gbmm12A	Tr12A	gbmm12A	Tr12A	gbmm12A	$\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$	$\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$	$\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$	$\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$
9.48	9.28	8.69	7.94	10.23	9.44	8.86	7.27	8.04	6.99	6.98	6.15

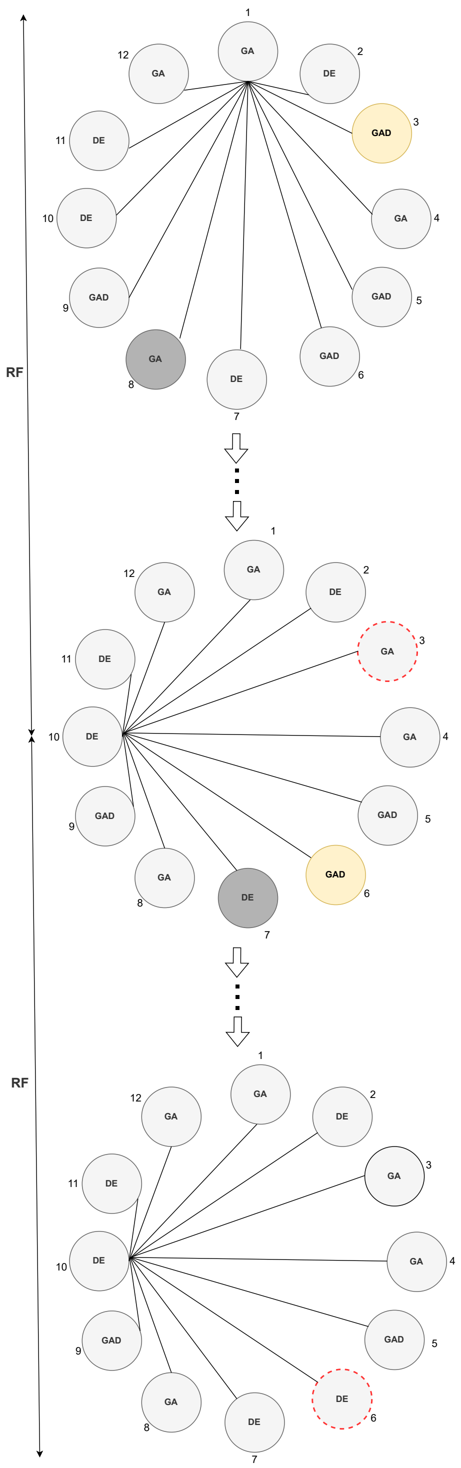


Figure 3.6: Example of periodic reconfiguration in the dynamic complete graph topology. For simplicity, only a subset of the edges between all islands is included. In each reconfiguration cycle, according to the number of generations controlled by the parameter *(RF)*, the gray island has the best performance, and the red-dotted island, with the worst performance, has its BA updated to the BA being executed by the gray island.

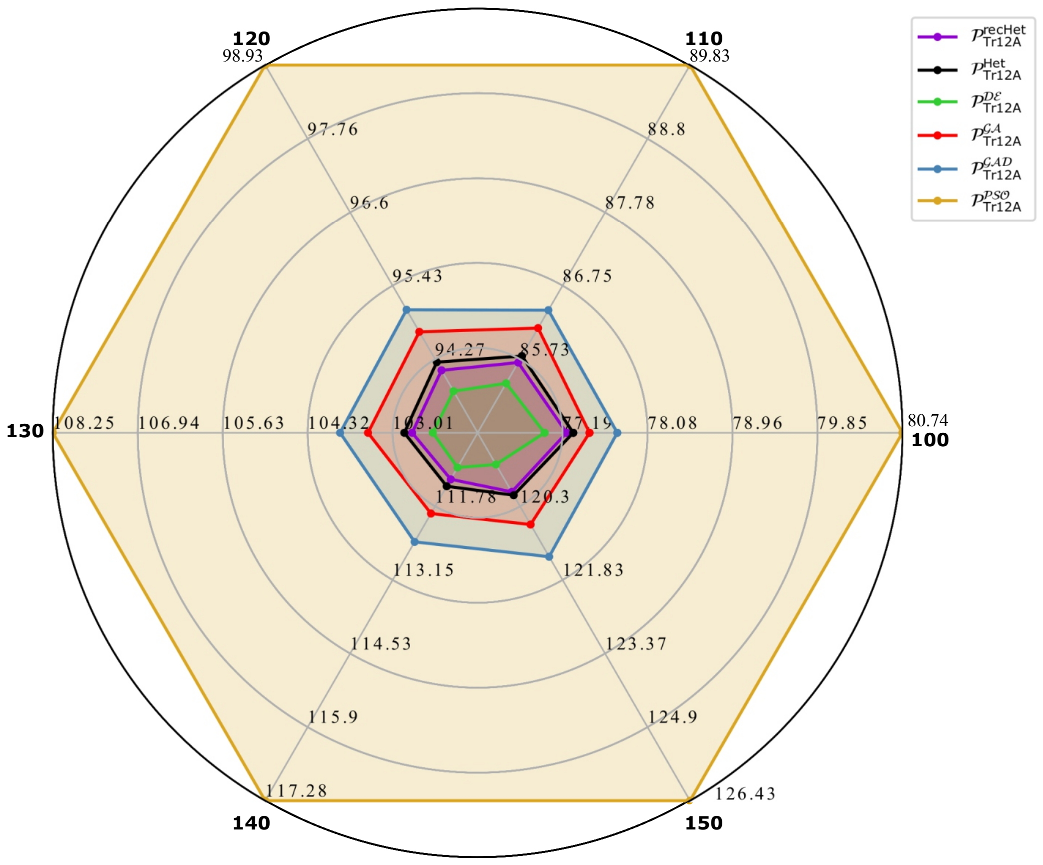


Figure 3.7: $\mathcal{P}^{\mathrm{recHet}}_{\mathrm{Tr12A}}$, $\mathcal{P}^{\mathrm{Het}}_{\mathrm{Tr12A}}$ and related HoPIMs accuracy.

3.6 Summary

This section serves as a summary to provide a comprehensive overview of the research outcomes, elucidating a mind map that encapsulates the published works resulting from the conducted research. We will further subdivide this section into two subsections for enhanced clarity and organization. In Subsection 3.6.1, we delve into publications centered around HoPIMs, delineating their contributions and outcomes. Subsequently, in Subsection 3.6.2, the focus shifts to specific publications related to HePIMs, offering insights into their unique characteristics and impact on the broader research landscape.

3.6.1 Summary on Homogeneous Parallel Island Models

Figure 3.9 presents a mind map associated with our research on HoPIMs, summarizing the evolution of our study discussed in sections 3.1 and 3.2. The nodes with the darker

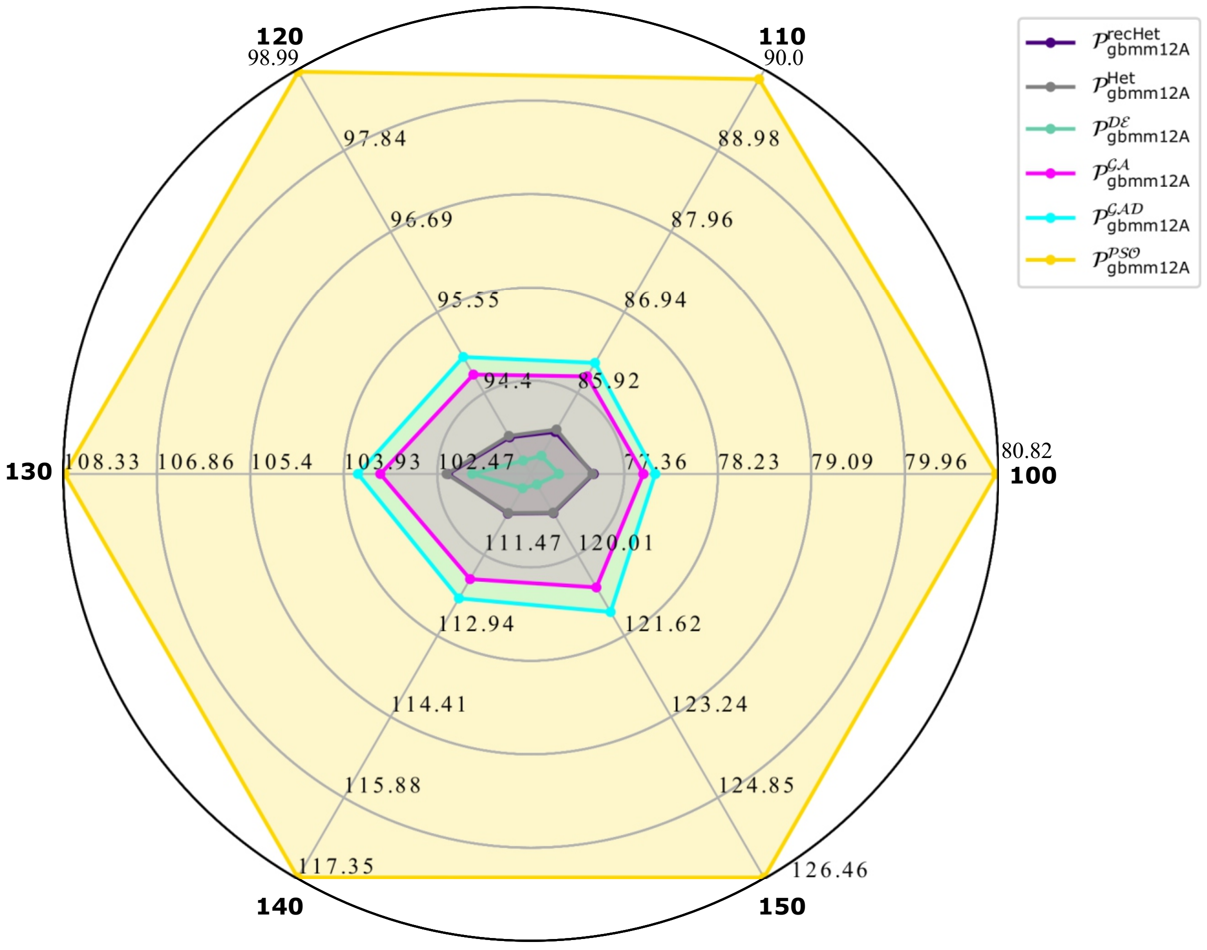


Figure 3.8: $\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$, $\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$ and related HoPIMs accuracy.

color represent the specific contributions of each work.

In the 2016 Congress on Evolutionary Computation edition, denoted in the mind map as **CEC2016**, we first explored the concept of PIMs. Using a static complete graph topology, we implemented a HoPIM from the `GA` with the UTD problem as a case study. The model adopted the same evolutionary parameters as the sequential version. It has an elitist migration policy where islands exchange their best individuals and replace their worst native individuals with immigrant individuals. However, the results of the experiments were not satisfactory, highlighting the need for accurate parameter tuning.

The following year, in **CEC2017**, the research continued with HoPIMs from the `GA`, utilizing neighborhoods constructed from static ring and complete graph topologies. Different population initialization strategies were explored, creating islands with random populations and others where the population was derived from the sequential version partitioned in equal sizes for each island. The URD problem was the case study, and we applied evolutionary and migration parameter tuning. HoPIMs provided the most effective solutions based on the ring topology, and the strategy of initializing islands from the population of the sequential version resulted in better solutions. It is important to note that the observation regarding the significance of parameter tuning in [12] was validated, and following this adjustment, all models delivered solutions superior to those computed with the sequential version.

In **CEC2018**, we designed HoPIMs, implemented from the `GA` to address the URD problem. Such models involve static topologies such as a binary tree, torus, and net. Additionally, we proposed a dynamic topology based on the islands' quality and diversity to build the neighborhood between islands during each migratory period. We employed an approach where each island generates its population randomly. A second approach involves dividing the population from the sequential version into equal sizes and then allocating it to the different islands of the model. From experiments, it was observed that the dynamic model yielded the best solutions. Additionally, partitioning the population from the sequential version showed superiority over creating random populations.

For the **CEC2019**, there was a shift to the UTD problem while maintaining static and dynamic topology HoPIMs from the `GA`, but adopting only initial populations created from the sequential version. Variations in the number of islands were introduced, with HoPIMs altered to operate with 12 and 24 islands while maintaining a total population of the same number of individuals. Each HoPIM underwent parameter adjustments, and the best results were obtained by models with 24 islands, particularly the dynamic model (*gbmm*).

We expanded our scope in **CEC2020**, adding static ring and complete graph topologies. We incorporated extra evolutionary algorithms like `SSA` and `PSO`, proposing variations with 12 and 24 islands for each algorithm (`GA`, `SSA`, `PSO`) and adjusting parameters. Regardless of the number of islands, all models had the same number of individuals. Initializing populations divided the global population from the sequential version equally among the islands. Experiments with the URD problem showed that GA-based models delivered the best solutions compared to models from `SSA` and `PSO`, respectively. However, models from the `SSA` proved poor solutions, unable to surpass the sequential version.

In the Journal on Applied Soft Computing, denoted in the mind map as **ASC2023**, the study exclusively employed the `GA` as the evolutionary engine. The focus was on evaluating both synchronous and asynchronous HoPIMs in various scenarios. Four case studies were considered: TMP, URD, URD, and N-Queens. Static and dynamic topologies with 12 and 24 islands were examined for synchronous and asynchronous models. All HoPIMs underwent parameter tuning for enhanced solutions. Experiments on the four distinct problems, considering the number of islands, synchronicity, and topology, resulted in good adaptation in terms of accuracy, with competitive speedups for all problems. The primary lesson reaffirmed that the best-adapted model is highly problem-dependent. It is essential to note that, for the same problem, small changes in parameters can have a significant combinatorial impact, directly influencing the best-adapted model. Although the emphasis is on accuracy, all HoPIMs provided substantial speed gains, confirming that PIMs are excellent strategies for addressing $\mathcal{NP}$-hard problems. It's worth highlighting that asynchronous HoPIMs offer better speedups only when all models have the same parameter configuration. The parameter selection performed in parameter tuning, aimed at achieving better accuracy results, has a direct impact on the execution speedup of a model.

3.6.2 Summary: Heterogeneous Parallel Island Models

Figure 3.10 presents the mind map related to the research on Heterogeneous Parallel Island Models, as described in works [14] and [15]. The node with darker coloring represents the specific contribution of this study.

Our research on heterogeneous models began at the Symposium Series on Computational Intelligence (SSCI) in 2021 [14] denoted as **SSCI2021**, where we delved into the fundamental concept in the literature regarding heterogeneous models. This concept involves dividing islands among various BAs, evolving them simultaneously, and exchanging individuals among them, as defined by migration parameters. We proposed a variety of synchronous and asynchronous HePIMs, utilizing: `GA`, `GAD`, `DE` and `PSO` as evolutionary engines for the islands, employing a static topology (Figure 2.6(a)) and another based on a dynamic complete graph (Figure 2.9). It's important to note that each algorithm had the same number of islands. As a case study, we used the URD problem and applied parameter tuning only to the migration parameters of each HePIM. The evolutionary parameters of the algorithms remained as defined through tuning for each HoPIM. The experiments demonstrated that asynchronous models deliver better solutions than synchronous ones. The inherent diversity in the asynchronous heterogeneous dynamic model was relevant in achieving the best results among all HePIMs. A comparison between heterogeneous and homogeneous versions revealed that heterogeneous models are superior, except for HoPIMs using `DE`, which proved superior to the heterogeneous ones. A lesson gleaned from this study is that overcoming HoPIMs is not as straightforward. When all islands operate synchronously, the island's evolution occurs with less diversity compared to asynchronous models.

At **SSCI2022**, we expanded upon the work initiated in [14]. We discarded synchronous heterogeneous versions as they provided inferior solutions compared to asynchronous HePIMs, retaining the evolutionary engines and the case study. We opted for a methodology called island reconfiguration, which involves fixed intervals defined as the number of executed generations, swapping the evolutionary engine of the worst-performing island with that of the best-performing island. The intervals are controlled by a parameter defined through parameter tuning. The results of the reconfiguration strategy evolved and narrowed the gap with the HoPIM from `DE`. Carefully analyzing the experiments, we observed a few points. The number of generations we defined, based on the input size, was not sufficient to converge the populations on the islands, meaning there was a chance to obtain better solutions if more evolutionary cycles were included. Another interesting observation is that islands executing PSO consistently showed significantly lower solution quality than the other islands, ultimately impacting the outcome. These insights are integrated into the evolution of the HePIMs published in [16] and discussed in the next chapter.

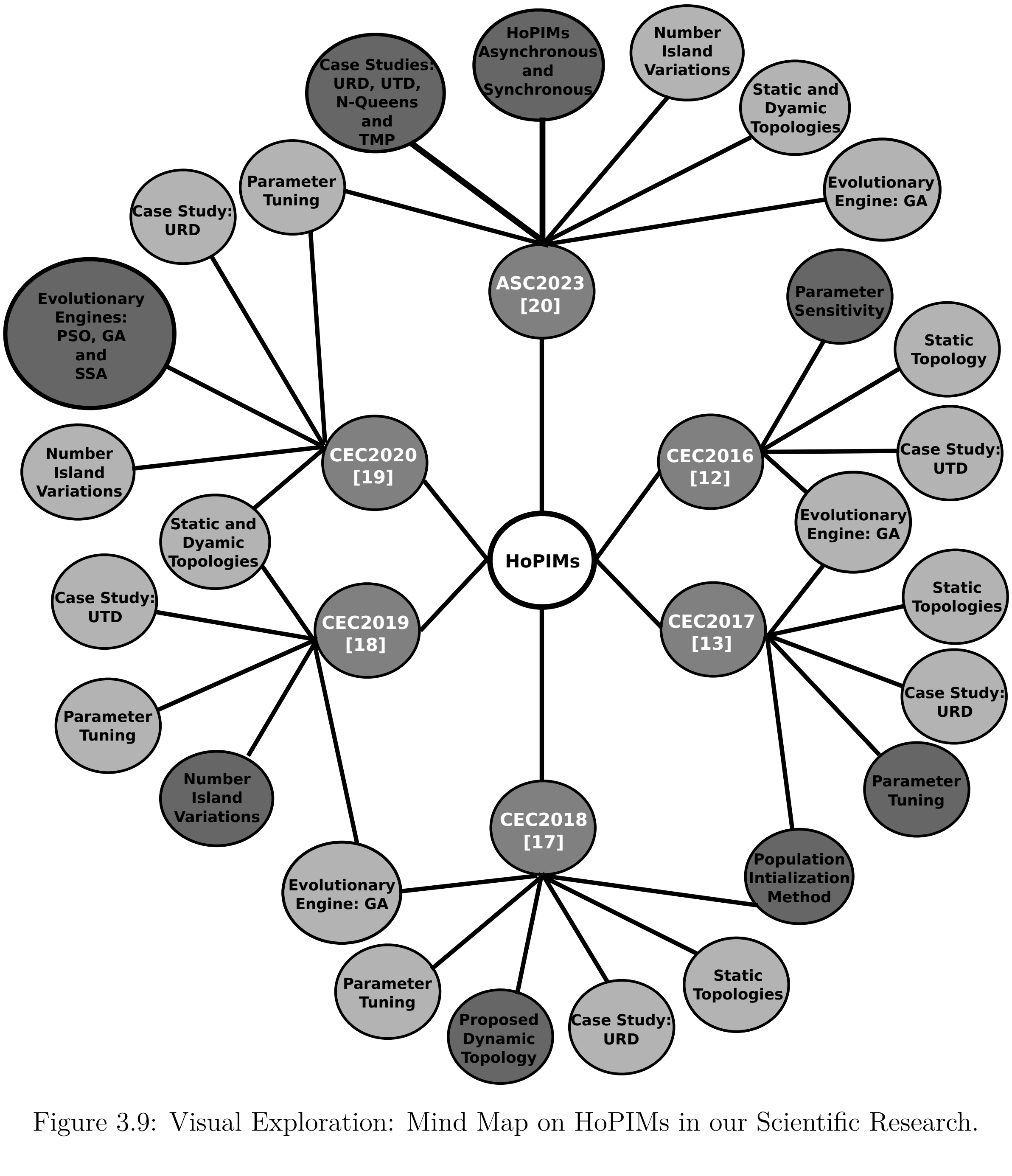


Figure 3.9: Visual Exploration: Mind Map on HoPIMs in our Scientific Research.

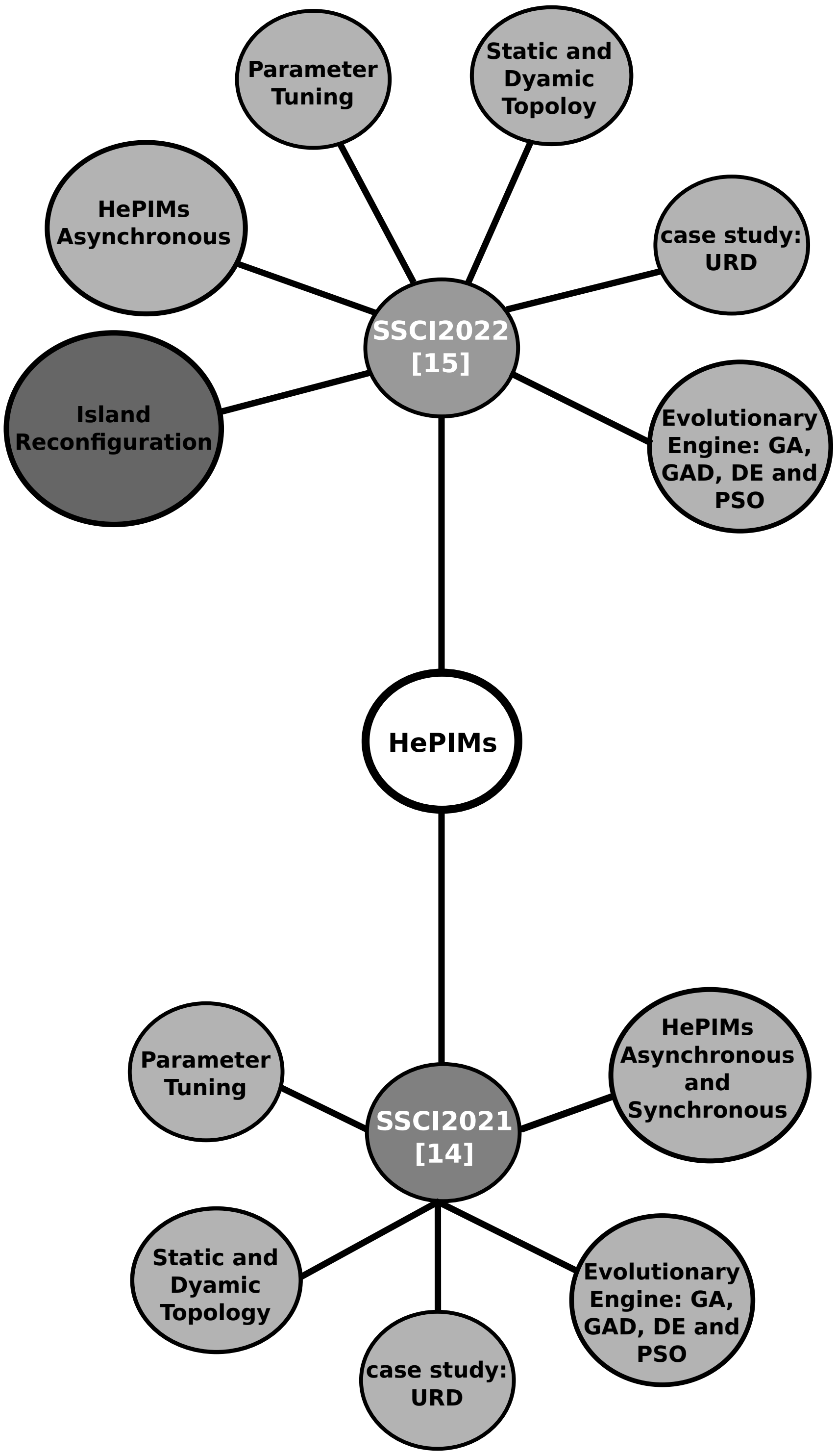

Parameter Tuning
Static and Dyamic Topoloy
HePIMs Asynchronous
case study: URD
SSCI2022 [15]
Island Reconfiguration
Evolutionary Engine: GA, GAD, DE and PSO
HePIMs
Parameter Tuning
HePIMs Asynchronous and Synchronous
SSCI2021 [14]
Static and Dyamic Topology
Evolutionary Engine: GA, GAD, DE and PSO
case study: URD

Figure 3.10: Visual Exploration: Mind Map on HePIMs in our Scientific Research.

Chapter 4

Stagnation-Based Reconfigurable Heterogeneous Parallel Island Models

4.1 Contribution

To improve the diversity and flexibility of HePIMs, to outperform the accuracy of results computed by the best-adapted HoPIMs (regarding [19, 14]), the feature of reconfigurability was first added to HePIMs in [15], obtaining encouraging and competitive results. In such models, each island may run a different BA and update it to the BA being executed by the island with the best performance, and the reconfiguration process was periodically performed after a fixed number of generations during the whole evolutionary process. However, the periodic reconfiguration policy proposed in [20] does not provide the required diversity to outperform the best-adapted HoPIMs. Results discussed in this chapter are published in [16].

This chapter introduces reconfigurable HePIMs with a so-called stagnation-based reconfiguration policy providing the required diversity. The algorithmic reconfiguration is applied continuously if island stagnation is detected. Such improvement in the dynamicity and flexibility of the reconfiguration phase is possible by maintaining a record of the algorithm executed by the most evolved island in a previous period of generations of the evolutionary process. The early reconfigurable PIMs proposed in [15] were refined after exhaustive experiments. Maintaining such a record eliminates the need to exchange information between the islands to decide how each island should reconfigure its BA.

The problem addressed is the URD. For comparison matters, we select the best-adapted HoPIMs to URD from [19] and the HePIMs that performed better from [14]. The HePIMs run three BAs: `GA`, `GAD`, and `DE`, and the best-adapted HoPIMs run `DE` in all its islands. Experiments are performed using two different topologies, a static binary tree topology and the dynamic complete graph topology proposed in [17].

The design decisions used by the stagnation-based reconfigurable HePIMs, introduced in this chapter, show that empowering HePIMs with the reconfiguration feature opens an exciting space for investigation since the accuracy of the computed solutions outperforms the results obtained by all previous PIMs. Moreover, the algorithmic convergence of stagnation-based reconfigurable PIMs is analyzed by implementing mechanisms to track island reconfiguration. The phase of the evolutionary process in which the model becomes homogeneous is identified. Interestingly, convergence does not always happen towards the best-adapted HoPIM as reported in [19, 14].

4.2 Reconfigurable HePIMs with stagnation policy

4.2.1 Communication Topologies

For comparison, the introduced stagnation-based reconfigurable models select one static and one dynamic topology that successfully addressed URD in [19, 15].

The static topology is a 12-island bi-directional binary tree. The dynamic topology is the 12-island complete graph (see Figure 4.1). Other 12- and 24-island topologies exhaustively examined in [19] were not considered since, in this work, we want to select the best-adapted PIMs to have a fair comparison.

In the complete graph topology, all pairs of islands may exchange individuals. Island communication dynamism is acquired by exploring the diversity and quality of each island, given by fitness variance and average metrics. Variance measures islands' diversity: high variance represents high population individuals' diversity, improving the chances of evolution into islands. The fitness average measures the quality of island populations. According to such metrics, the islands are ranked as **good**, **bad**, and **medium**. Migrations exchange individuals between **good** and **bad** islands, and **medium** and **medium** islands only (for short, `gbmm`).

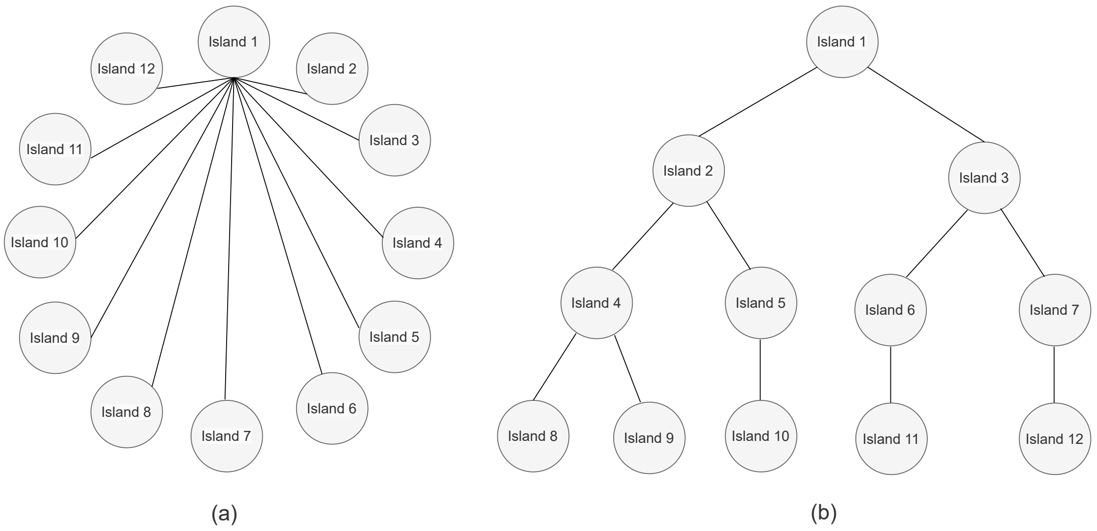


Figure 4.1: (a) A dynamic complete graph topology. (b) binary tree topology. In (a), only a subset of the edges between all islands is included for simplicity.

4.2.2 Reconfigurable islands

Reconfigurable HePIMs were proposed initially in [15]. The dynamic complete graph model classifies islands according to their fitness average and variance and uses it to build neighborhoods during the evolutionary process. A master island is responsible for receiving data from all islands; it ranks and notifies the worst island. The worst island reconfigures (i.e., replaces) its algorithm with the algorithm of the best-ranked island. Furthermore, the reconfiguration process is controlled by a parameter called reconfiguration frequency *(RF)*. The *RF* establishes the number of evolutionary generations to (periodically) perform a reconfiguration step. The dynamic reconfiguration allows updating the BAs executed in their islands. In [15], the authors implemented two reconfigurable HePIMs: $\mathcal{P}^{\mathrm{recHet}}_{\mathrm{Tr12A}}$ and $\mathcal{P}^{\mathrm{recHet}}_{\mathrm{gbmm12A}}$ selected as the most competitive HePIMs from [19]. The former uses the static binary tree topology, and the latter uses the dynamic complete graph topology; both models are asynchronous and evolve through a refined migration policy that allows the exchange of individuals, maintaining diversity.

4.2.3 Stagnation-based Reconfigurable HePIMs

In the stagnation-based reconfiguration approach, instead of conducting a reconfiguration phase after a fixed number of generations (determined by the parameter RF), the new model may reconfigure the current BA in each island to the best BA in all evolutionary generations (See Figure 4.2). The best BA is the BA used by the island, showing the highest progress in a fixed interval of generations. This interval is given by a BA classification interval ($BACI$) parameter. Exhaustive experiments showed that a good metric to evaluate the progress of the islands is granted by the difference between the best individual fitness at the beginning and the end of the interval. Each island maintains a record of its progress in the current and previous three generations during the evolutionary process. The island is reconfigured to continue the evolutionary process, executing the best BA whenever no progress is detected. In contrast to the strategy of periodic reconfiguration, the island's ability to maintain this record of progress permanently makes reconfiguring any *stagnated* island possible in each generation.

Furthermore, since the best BA is not known during the initial $BACI$ classification interval, reconfigurations only start afterward. Once the first phase of the classification of BAs is performed, stagnated islands will continuously update their current BA to the best BA. The design decision to fix island stagnation as not improving the best individual fitness in the last three generations was taken according to exhaustive experiments in which initially reconfiguration was performed periodically in shorter intervals than those applied in [15], and secondly, in which each island was able to reconfigure its BA during all generations according to its performance in different evolutionary generation intervals.

The flowchart in Figure 4.3 illustrates the abstraction of the model. Initially, a global population is equally distributed among the islands, with the same number of generations assigned to each island. Although we explicitly present only islands one and twelve for spacing reasons, the procedure is analogous for islands two to eleven. Each island executes the initial 10% of generations without modifying the evolutionary algorithm. Subsequently, the stagnation-based reconfiguration mechanism is activated. After each interval of 10% of the total number of generations, all islands send information to island one, responsible for calculating which island evolved the most during the interval. Island one, in turn, sends the algorithm of the chosen best-performing island to all other islands. This process allows any island to replace its current by the best algorithm when it stagnates. The flowchart also encompasses the migration process, as islands need to exchange individuals at some point, considering the topology used to form neighborhoods. We do not delve into specific topology details here; consider it dynamic or static. Upon completing the generations, the individual with the highest fitness among all islands is deemed the solution. The new stagnation-based reconfigurable HePIMs are denoted according to their topology as $\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$.

4.3 Experiments and analysis of accuracy

All PIMs, including the new reconfigurable models, were implemented using the `MPI` library of `C` in Linux, and for the sake of comparison, experiments were executed on a computational platform using two Xeon E5-2620 2.4 GHz six-core processors with hyper-threading.

To adapt `DE` to URD, each n-dimensional individual v is represented by a numerical vector with values in the continuous interval $[0, 1]$, which is associated with the permutation $\pi = (\pi_1, \ldots, \pi_n)$ given as input. The individual associated to v is an oriented version

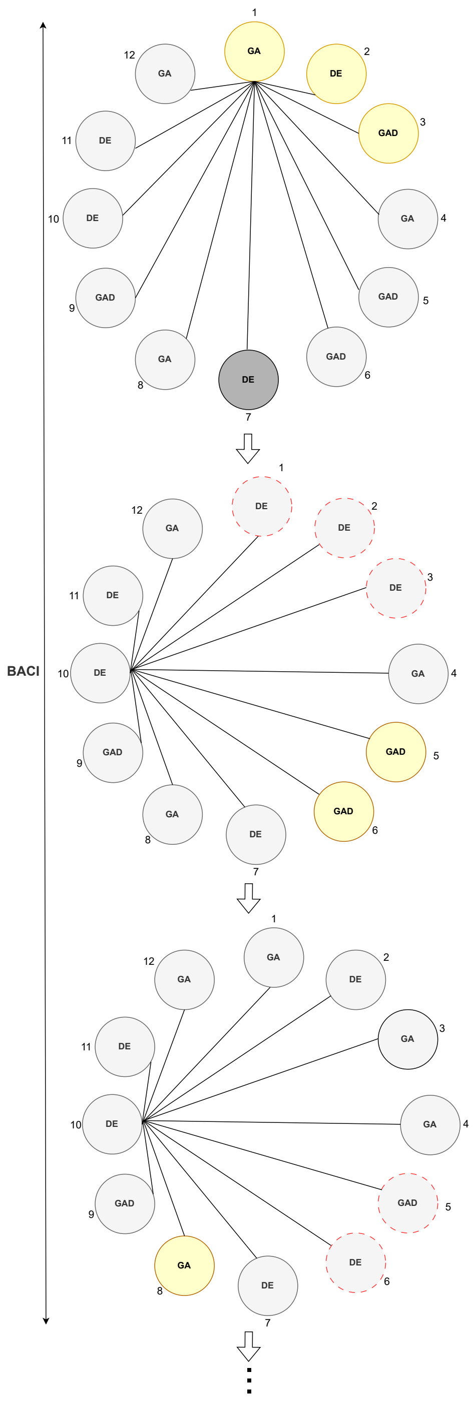


Figure 4.2: Example of stagnation-based reconfiguration on the complete graph topology. For simplicity, only a subset of the edges between all islands is included. The yellow islands represent stagnated islands in each generation that have undergone reconfiguration to the BA algorithm executed by the best performance island, in gray, computed at the beginning of each cycle of evolutionary generations defined by the parameter BACI.

of π such that for all i, if the i-th entry of v belongs to the interval $[0, 0.5)$ then π_i receives a negative orientation: $\overleftarrow{\pi_i}$; otherwise, if it belongs to the interval $[0.5, 1]$, π_i is assigned

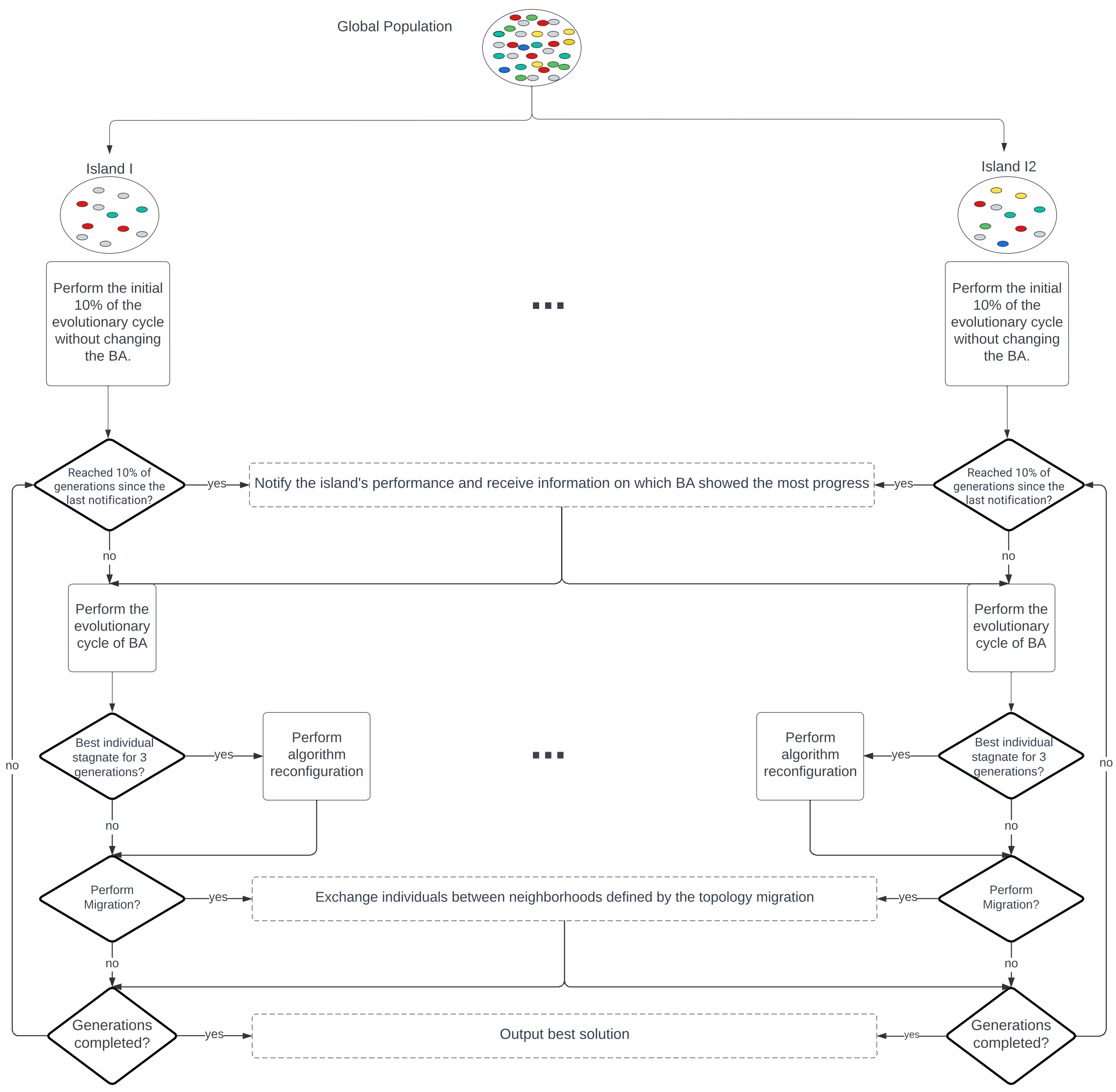

Global Population
Island I
Island I2
Perform the initial 10% of the evolutionary cycle without changing the BA.
Perform the initial 10% of the evolutionary cycle without changing the BA.
Reached 10% of generations since the last notification?
yes
Notify the island's performance and receive information on which BA showed the most progress
yes
Reached 10% of generations since the last notification?
no
no
Perform the evolutionary cycle of BA
Perform the evolutionary cycle of BA
Best individual stagnate for 3 generations?
yes
Perform algorithm reconfiguration
Perform algorithm reconfiguration
yes
Best individual stagnate for 3 generations?
no
no
no
no
Perform Migration?
yes
Exchange individuals between neighborhoods defined by the topology migration
yes
Perform Migration?
no
no
Generations completed?
yes
Output best solution
yes
Generations completed?

Figure 4.3: Flowchart of the stagnation-based reconfigurable HePIMs.

positively: $\overrightarrow{\pi_i}$. In this manner, a signed permutation is built from v and π.

The basis for comparing the performance of PIMs are sequential versions of `GA`, `GAD` and `DE` with populations of size $24n \log n$, for inputs of length n, and 200 generations. Previous experiments were performed considering n generations for inputs of length n, where 150 was the longest input length. Empirically, we realize that increasing the number of generations almost closed the accuracy gap between the best-adapted HoPIM, HePIMs, and reconfigurable HePIMs in [15]. Therefore, the decision is to increase the number of generations for all inputs to 200. Such many generations are also relevant to observing how stagnation-based reconfigurable HePIMs behave and evolve during the evolutionary process regarding the best-adapted models (cf. bar diagrams in Figures 4.7, and 4.12)

For a fair comparison, all PIMs deal with populations of the same size. Also, we select three static binary three and three dynamic complete graph 12-island asynchronous HoPIMs, designed in [19], each running one of the BAs: `GA`, `GAD` and `DE`. In addition, two asynchronous 12-island HePIMs, with the topologies of the HoPIMs above, presented in [14], were adjusted, running in their islands the BAs: `GA`, `GAD` and `DE`. Finally, two asynchronous 12-island reconfigurable HePIMs, with identical topologies as above, and applying (periodic) reconfiguration at fixed generation frequencies were adapted from [15]. It is important to remark here that in [19, 14] also, 24-island HePIMs and different topologies were designed and verified. However, the best models from these papers were selected for a fair comparison with the novel reconfigurable HePIMs.

The HoPIMs are $\mathcal{P}^{\texttt{GA}}_{\text{Tr12A}}$, $\mathcal{P}^{\texttt{GAD}}_{\text{Tr12A}}$, $\mathcal{P}^{\texttt{DE}}_{\text{Tr12A}}$, $\mathcal{P}^{\texttt{GA}}_{\text{gbmm12A}}$, $\mathcal{P}^{\texttt{GAD}}_{\text{gbmm12A}}$ and $\mathcal{P}^{\texttt{DE}}_{\text{gbmm12A}}$. The superscripts denote the BA used by the homogeneous model. The subscript prefixes indicate whether the model uses the static tree (`Tr`) or the dynamic complete graph topology (`gbmm`), and the subscript suffix `12A` indicates the number of islands and that the model is asynchronous. Furthermore, it is essential to point out that the homogeneous model $\mathcal{P}^{\texttt{DE}}_{\text{gbmm12A}}$ provides the best solutions for the URD problem.

The HePIMs are $\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$ proposed in [14], and the reconfigurable HePIMs with fixed (periodic) reconfiguration frequency are $\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$ proposed in [15]. The new HePIMs with stagnation-based reconfiguration are $\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$. All PIMs have the following configuration:

- Each island has $2n \log n$ individuals, for inputs of length n;
- Initially, islands 1, 4, 8 and 12 run `GA`, islands 2, 7, 10 and 11 `DE`, and islands 3, 5, 6 and 9 runs `GAD`;
- Generation number is fixed at 200.

4.3.1 Parameter Setup

We use the parameters for BAs, HoPIMs, and HePIMs obtained in [14] and [15] after an exhaustive parameter setting process. The *parameter tuning* adopted the "taxonomy T1" in [90]. In [15], the reconfiguration period of reconfigurable HePIMs was calibrated after exhaustive experiments; this is the unique additional parameter allowed for reconfigurable HePIMs. Similarly, for the new stagnation-based reconfigurable HePIMs, the number of three generations to define island stagnation was determined after exhaustive experiments. Indeed, for other elaborated reconfigurable HePIM designs, which eventually consider different BAs, topologies, and migration policies, all parameters, including the reconfiguration period and (number of generations for) stagnation, should be subjected to parameter calibration. Table **??** presents the parameter ranges. For percentages, the tested values range between 2% and 100%. For probabilities, the values range from 0.02 to 1.0, and for the mutation parameter, from 0.01 to 0.02. As for the *mutation* probability parameter, bounded by $0,02$, for `DE`, the F_M percentage parameter is set in the range of 1% to 2%. The upper bound of 2% was defined based on the analysis of our parameter adjustment process, in which solutions with F_M greater than 2% substantially degrade the quality of the solutions.

Table 4.2 presents the parameter value configuration for sequential versions and HoPIMs, while Table 4.3 shows the parameter values for HePIMs.

4.3.2 Analysis of Accuracy

The experiments were conducted as described below, considering preliminary results on reconfigurable models obtained in [15].

- The number of generations was extended to 200 for all samples to guarantee the desired effect from the stagnation-based reconfigurable approach along the evolutionary process. In previous works, the number of evolutionary generations was

Table 4.1: Parameter Value Ranges

	Parameter	Parameter values
`GA` and `GAD`	*crossover*	$0.02, 0.04, \cdots, 0.98, 1.0$
	mutation	$0.01, 0.011, \cdots, 0.019, 0.02$
	selection	$2\%, 4\%, \cdots, 98\%, 100\%$
	replacement	$2\%, 4\%, \cdots, 98\%, 100\%$
`DE`	P_C	$0.02, 0.04, \cdots, 0.98, 1.0$
	F_M	$1\%, 1.1\%, \cdots, 1.9\%, 2\%$
Migration	*IN*	1,2,3,4,5,6,7,8,9,10,11,12,13
	EMI	1=Best, 2=Worst, 3=Random
	EP	1=Clone, 2=Remove
	IMI	1=Worst, 2=Random, 3=Similar
	MI	$2\%, 4\%, \cdots, 98\%, 100\%$

Table 4.2: Parameter Settings for `GA`, `GAD`, `DE` and associated HoPIMs.

Parameter	`GA`	$\mathcal{P}^{\texttt{GA}}_{\text{Tr12A}}$	$\mathcal{P}^{\texttt{GA}}_{\text{gbmm12A}}$	`GAD`	$\mathcal{P}^{\texttt{GAD}}_{\text{Tr12A}}$	$\mathcal{P}^{\texttt{GAD}}_{\text{gbmm12A}}$	`DE`	$\mathcal{P}^{\texttt{DE}}_{\text{Tr12A}}$	$\mathcal{P}^{\texttt{DE}}_{\text{gbmm12A}}$
crossover	0.90	0.98	0.96	0.92	0.98	0.98			
mutation	0.02	0.015	0.011	0.01	0.01	0.01			
selection	60%	92%	94%	98%	98%	94%			
replacement	60%	70%	70%	90%	80%	90%			
P_C							0.74	0.72	0.78
F_M							1%	1.4%	1%
IN		9	5		12	5		3	5
EMI		1	1		1	1		1	1
EP		2	2		2	1		1	2
IMI		1	1		1	1		1	1
MI		30%	30%		14%	12%		14%	12%

Table 4.3: Parameter Settings for HePIMs.

Parameter	$\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$	$\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$	$\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$	$\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$	$\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$	$\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$
IN	3	6	3	6	3	6
EMI	1	3	1	3	1	3
EP	2	2	2	1	2	1
IMI	3	3	3	3	3	3
MI	10%	10%	10%	14%	10%	14%
RF			14%	24%		
BACI					10%	10%

fixed as the length of the permutation inputs reaching the maximum number in samples of length 150 evolved during 150 generations (cf [14, 15]).

- The `PSO` algorithm, used by HePIMs in previous work, is no longer used as an evolutionary engine because it consistently performs much worse than algorithms: `GA`, `GAD` and `DE`. When considering heterogeneous versions, the `PSO` maintained a slice of three islands, and due to its inferior performance, it negatively impacted the solutions of heterogeneous PIMs (see [15]). Also, the social spider algorithm was discarded based on its restricted performance regarding self-adjusting PSO over static and dynamic HoPIMs as investigated in [19].
- We diversified the length of the input instances:
  - For each permutation length, $n \in \{100, 110, \ldots, 150\}$, one package of one hundred unsigned permutations with n genes was randomly generated;
  - All PIMs were executed ten times (a total of one thousand executions) using each of the permutations of length n, and the average of these executions for each permutation was taken as the result. The average gives the computed number of reversals for each unsigned permutation.

The radar chart in Figure 4.4 compares the sequential algorithms `GA`, `GAD`, and `DE`. Since URD is a minimization problem, the smaller the output, the higher the accuracy. The radar chart shows that `DE` presents the best and `GA` the worst solutions. The algorithm `GAD` only computes better solutions than `DE` for inputs of length 130. The genetic algorithm variants `GA` and `GAD` behave differently as the search space increases: for permutations of lengths 100 and 110, `GA` provides the best solutions, but the scenario is reversed for longer inputs.

The radar charts in Figure 4.5 and the radar chart in Figure 4.6 show results from the HoPIMs. They show how sequential versions are easily overcome by HoPIMs and that the accuracy of the outputs computed by dynamic HoPIMs is better than the outputs

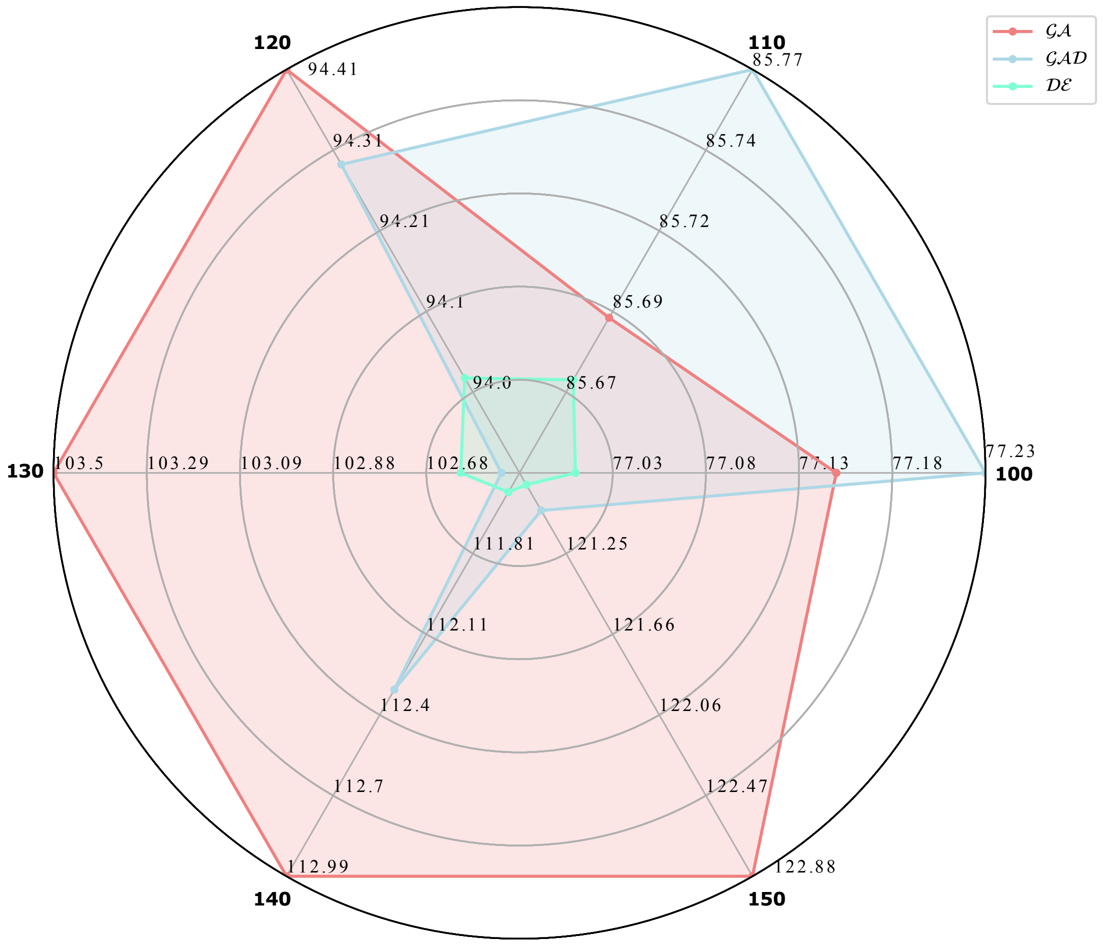
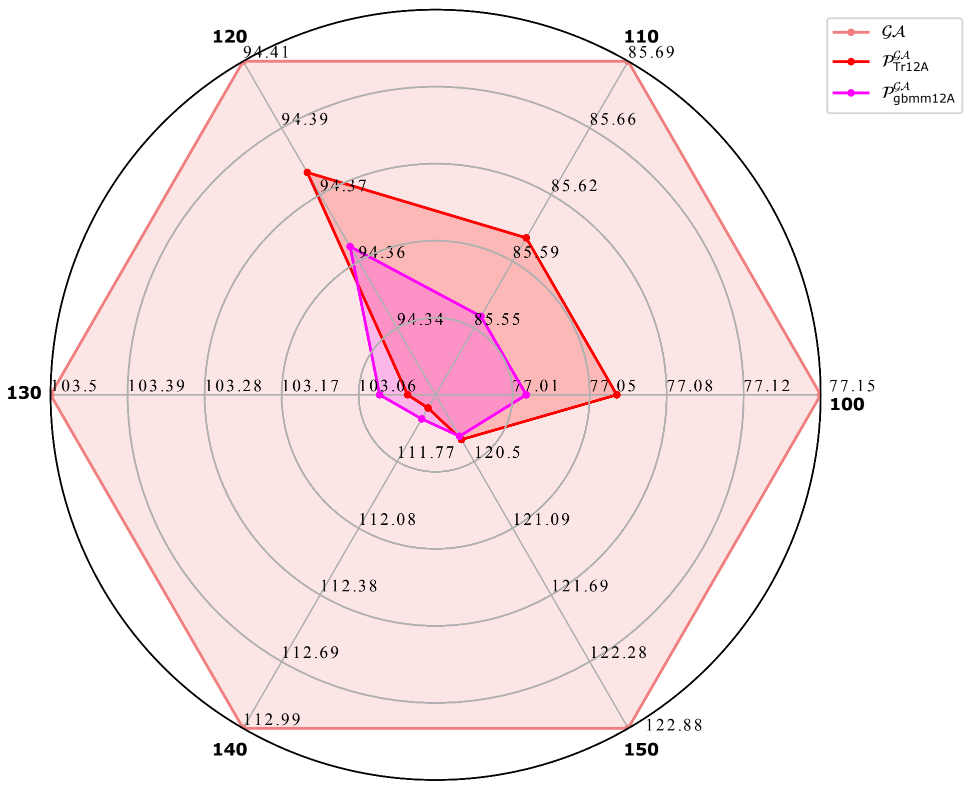


Figure 4.4: Accuracy of the sequential BAs: GA, GAD and DE. The radar chart is scaled according to the worst performance for each input size. Since the target optimization problem URD is a minimization problem, the smaller the radius, the better the result.

computed by static versions. The three radar charts have different scales, and for comparison, the radar chart in Figure 4.8 presents all static and dynamic homogeneous PIMs. The best-adapted model is the dynamic HoPIM with BA DE, $\mathcal{P}^{\mathtt{DE}}_{\mathrm{gbmm12A}}$. The strength of dynamic models has already been presented in previous work. The property of not having a fixed neighborhood allows faster dissemination of genetic material with high evolutionary potential among the islands. A less dominant scenario is obtained with experiments involving HoPIM from the GA as seen in the above radar chart in Figure 4.5, where only for inputs of size $100, 110$, it is possible to see a substantial improvement of the dynamic HoPIM.

The above and below radar charts in Figure 4.9 compare the results of the static



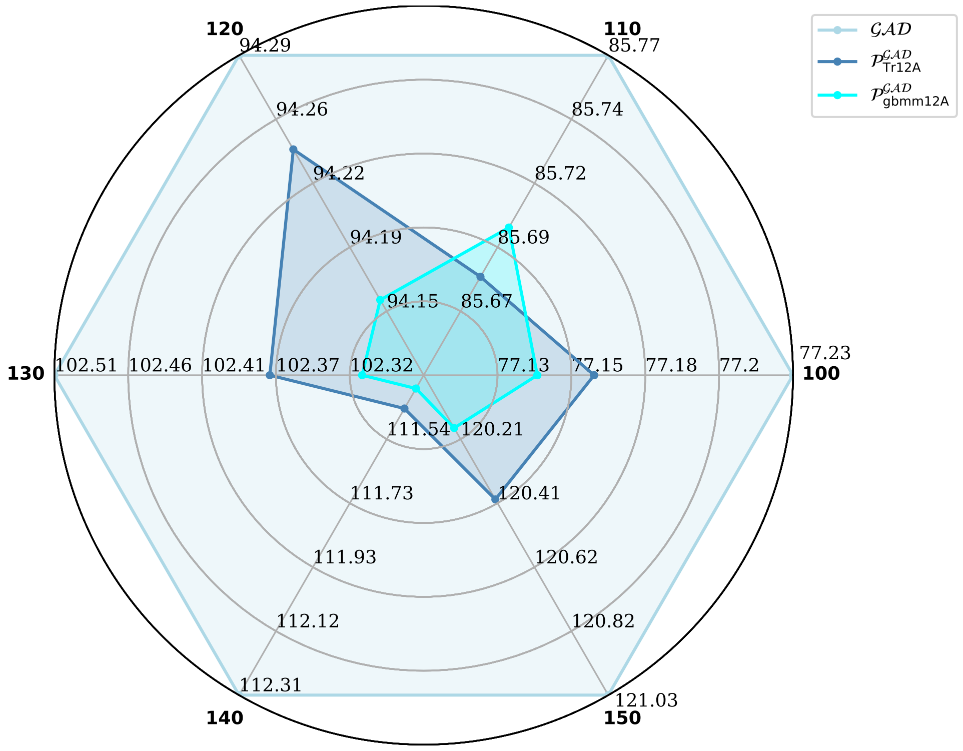


Figure 4.5: Above radar chart: accuracy of the HoPIMs from `GA`. Below radar chart: Accuracy of the HoPIMs from `GAD`. Each radar chart is scaled according to the performance of the associated sequential algorithm since it provides the worst performance.

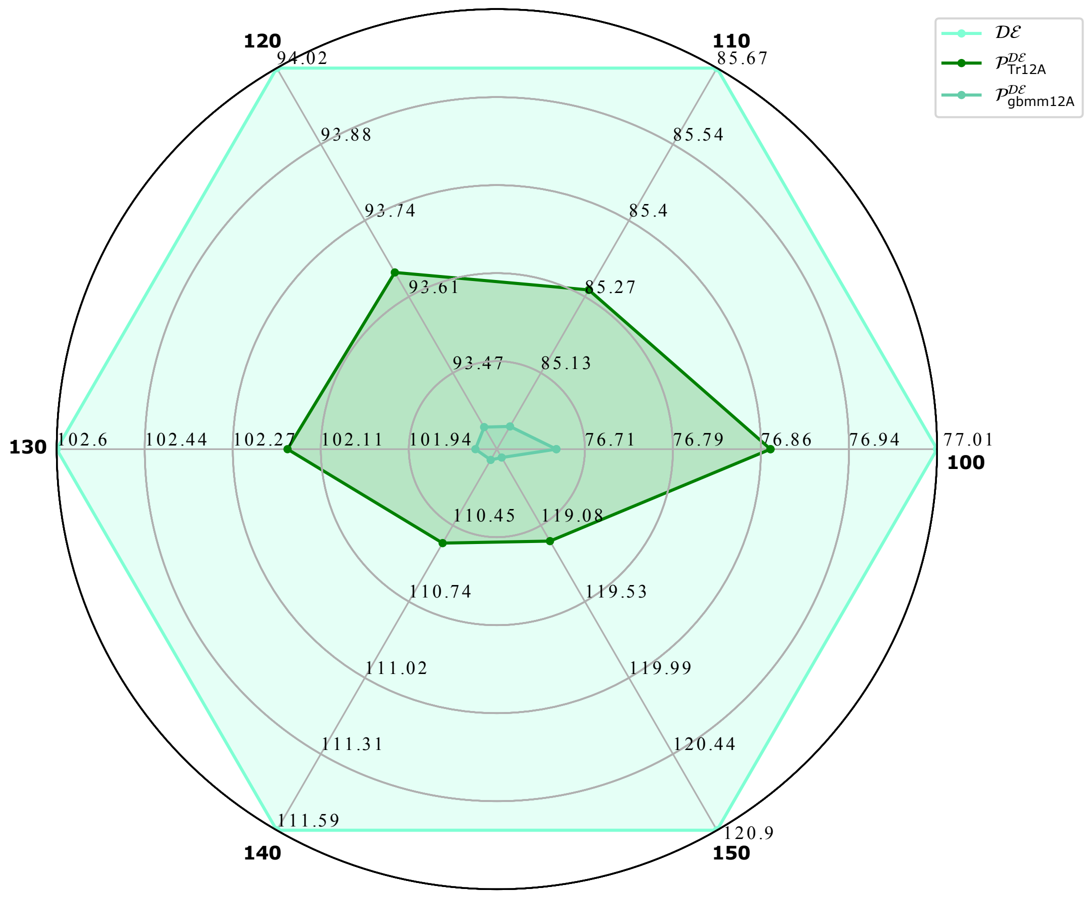


Figure 4.6: Above radar chart: accuracy of the HoPIMs from DE. Below radar chart: Comparison of the accuracy of all HoPIMs. The first radar chart is scaled according to the accuracy of the sequential model, while the second radar chart is scaled according to the model with the worst accuracy for each input length.

and dynamic HePIMs versus the best static and dynamic HoPIMs, $\mathcal{P}^{\text{DE}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{DE}}_{\text{gbmm12A}}$, respectively. Regarding static models (see above chart), the HoPIM presents the worst solutions. The static neighborhood scenario imposed by the binary tree topology benefits HePIMs because islands always send and receive individuals with different evolutionary characteristics, which favors not being trapped in local optima. The static HePIMs show very similar behavior, with the proposed stagnation-based reconfigurable model $\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$ not performing better than the competing periodic reconfigurable model $\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$ and the heterogeneous model, $\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$. Considering longer instances of the URD problem, the periodic reconfigurable model, $\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$, is the best static HePIM. On the other hand, the stagnation-based reconfiguration method succeeded for the dynamic topology (see the

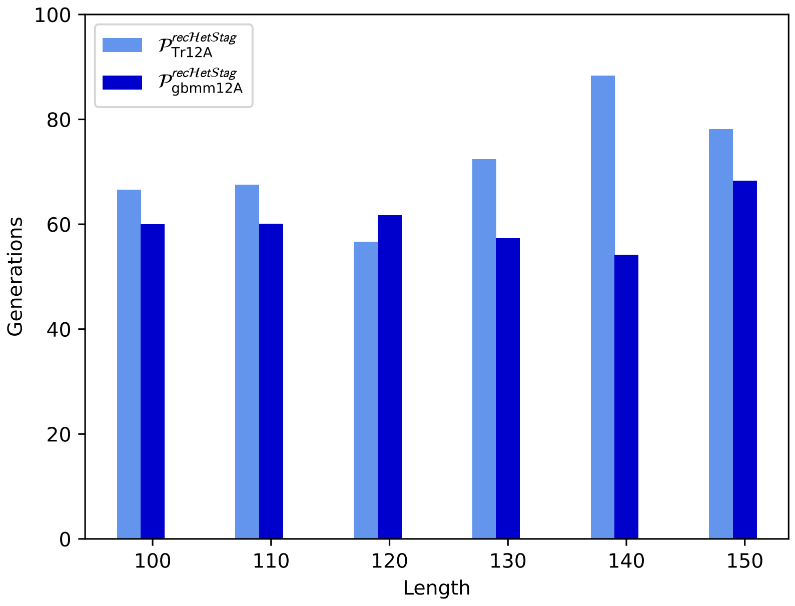


Figure 4.7: The diagram shows the number of generations required for the reconfigurable HePIMs $\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ to become HoPIMs, i.e., to run the same BA in all their islands homogeneously.

below chart); indeed, $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ presented the best quality solutions, and the distance to the second best model ($\mathcal{P}^{\text{DE}}_{\text{gbmm12A}}$) becomes even more significant when considering large instances of the URD problem.

The radar chart in Figure 4.10 compiles all results in both radar charts in Figure 4.9 using the same scale. In this radar chart, it is clear how the migration policy on the dynamic topology of the complete graph model jointly with stagnation-based reconfiguration approach provided by the model $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ give the required diversity to outperform all other heterogeneous and reconfigurable heterogeneous, and the best adapted homogeneous models.

A point that will arouse the reader's curiosity about the reconfiguration method is how the islands end up when all generations are executed. After the evolutionary cycle, both $\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ have all the islands running the same BA, and both become homogeneous. The average number of generations required for the models to become homogeneous is shown in Figure 4.7. The $\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$ needs more generations until it becomes

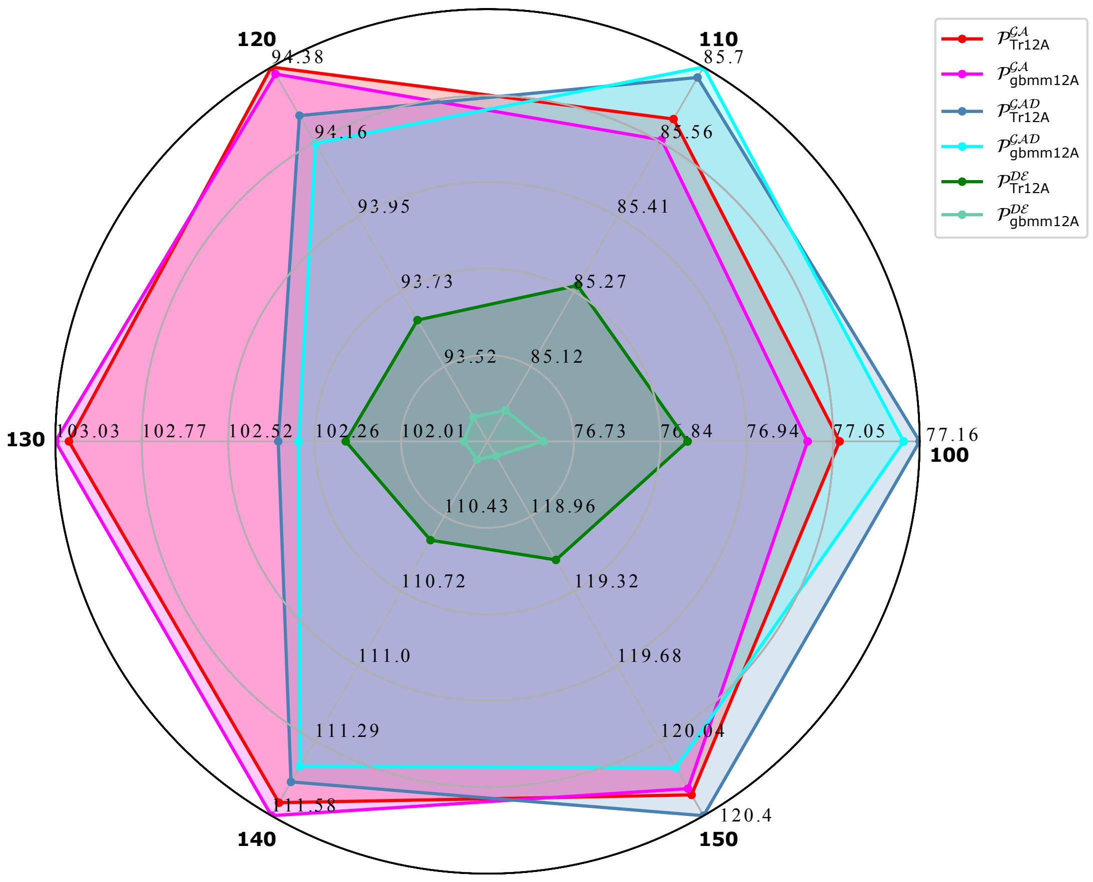


Figure 4.8: Above radar chart: accuracy of the HoPIMs from `DE`. Below radar chart: Comparison of the accuracy of all HoPIMs. The first radar chart is scaled according to the accuracy of the sequential model, while the second radar chart is scaled according to the model with the worst accuracy for each input length.

homogeneous compared with the $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$, except for the sample size 120. Also, regardless of the model, the reconfiguration process is complete before reaching 100 generations.

Another question is which BA is dominant? Investigating the input samples for both stagnation-based reconfigurable models, $\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ we have the scenario presented in Table 4.4. Only `DE` and `GA` are being shown because there were no samples where the islands ended running the algorithm `GAD`. The values in the table are the percentage of inputs ending with all islands executing either the algorithm `DE` or `GA`. The algorithm `DE` is dominant for all samples. In addition, the fact that the algorithm `GAD` never has dominance is unusual since, as shown in Figure 4.4, it delivers better quality solutions than the algorithm `GA`. We investigated it and realized that the HoPIMs from the `GA` present better solutions than `GAD` if we consider a maximum of 150 generations (see Figure 4.11). Furthermore, as seen in Table 4.4, before 100 generations, all islands have already become homogeneous. We can attribute the success of model $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ to the aptitude of the dynamic topology and the fact that it becomes homogeneous earlier. Hence, the evolutionary engine DE has more time to evolve a population that already has individuals shaken by GA and GAD, which makes it difficult to stay stuck in great locations.

Finally, partial accuracy results of $\mathcal{P}^{\texttt{DE}}_{\text{gbmm12A}}$ and $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ during the evolutionary cycle are compared in Figure 4.12. The experiment is performed with inputs of length 150. We use the accuracy population average to measure island development at intervals of 20 generations. The dynamic stagnation-based reconfigurable model $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ produces populations better adapted to the URD problem regardless of the analyzed interval. Although the partial accuracy evolution between both models is always very close, it is noticeable that the advantage caused by heterogeneity and algorithmic stagnation-based reconfiguration regarding the best adapted homogeneous model, $\mathcal{P}^{\texttt{DE}}_{\text{gbmm12A}}$, at the beginning of the evolutionary cycle is maintained and even improved after the model becomes homogeneous (cf. Figure 4.7).

Table 4.4: Configuration of islands of the reconfiguralbe HePIMs $\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ at the end of the evolutionary cycle (200 generations).

Length	$\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$		$\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$	
	DE	GA	DE	GA
100	90%	10%	100%	
110	100%		80%	20%
120	100%		100%	
130	90%	10%	90%	10%
140	100%		70%	30%
150	90%	10%	90%	10%

4.3.3 Performance

The speed-up of the homogeneous models $\mathcal{P}^{\texttt{DE}}_{\text{Tr12A}}$ and $\mathcal{P}^{\texttt{DE}}_{\text{gbmm12A}}$ was evaluated regarding the runtime of the sequential algorithm DE. The input dataset with permutations of length 150 was chosen because such permutations represent more challenging, inherently

Table 4.5: Speed-up for HePIMs regarding the sequential version of GA, GAD, and DE for the dataset with genomes of length 150. The table also includes the average time in seconds of each algorithm (in the second column and second row).

		$\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$	$\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$	$\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$	$\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$	$\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$	$\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$
	sec	7.771	8.126	9.761	8.100	8.873	10.120
GA	52.051	6.698	6.405	5.333	6.426	5.866	5.143
GAD	49.765	6.404	6.124	5.098	6.144	5.609	4.917
DE	59.752	7.689	7.353	6.122	7.377	6.734	5.904

complex problems than shorter permutations. The algorithms $\mathcal{P}^{\texttt{DE}}_{\text{gbmm12A}}$, $\mathcal{P}^{\texttt{DE}}_{\text{Tr12A}}$, and DE were executed ten times for each permutation in the dataset, and the meantime is the average runtime. The dynamic homogeneous model $\mathcal{P}^{\texttt{DE}}_{\text{gbmm12A}}$ reached a speed-up of 7.18 while the static homogeneous model $\mathcal{P}^{\texttt{DE}}_{\text{Tr12A}}$ a speed-up of 8.37.

Due to the nature of HePIMs, where groups of islands run different BAs, it is not easy to decide which sequential BA would be the reference to compute the speed-up; thus, the Table 4.5 shows the speed-up of heterogeneous models concerning the BAs GA, GAD, and DE. The methodology to compute speed-ups for HePIMs is the same as that applied to HoPIMs. The simple versions of HePIMs provide the best speed-ups. Static HePIMs deliver better performances than dynamic HePIMs since dynamic topology PIMs build neighborhoods for the islands at each migratory cycle. The proposed reconfiguration process involves sophisticated techniques that improve the quality of the solutions; however, it impacts performance. The stagnation-based reconfigurable HePIMs $\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ are slower than the periodic reconfigurable models $\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$ and $\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$ since they make use of an effortless reconfiguration process.

4.3.4 Statistical Analysis

We apply Friedman's test and Holm's analysis. Applying such a statistical methodology is motivated by the discussion in [91], which guarantees its application in our case, where multiple comparisons should be applied over discrete populations, complemented with a further post hoc analysis to determine which algorithms differ from the others. Statistical tests validated experiments with a significance level $\alpha = 0.05$, i.e., 5% of chance of incorrectly rejecting the null hypothesis in the long term.

The samples are the sets of one hundred outputs of lengths from 100 to 150 genes considered in Section 4.3.2. The first step of the analysis consists of applying the Friedman non-parametric statistical test to define the control algorithm. Then, the multiple-hypotheses Holm's testing method is applied to check the null hypothesis that the performance of the control algorithm is the same as that of the remaining algorithms.

Table 4.6 presents the statistical results for the HePIMs and the best-adapted HoPIM, $\mathcal{P}^{\mathsf{DE}}_{\text{gbmm12A}}$. In the first step of the analysis, the Friedman test selects $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ as the control algorithm. In the second step, Holm's method rejects the null hypotheses for p-value ≤ 0.05. In Table 4.6, an algorithm has statistical significance, whenever p-value $\leq \alpha/i$. The table shows that the selected model, $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$, always has statistical significance for all other HePIMs regardless of the input length. On the other hand, when $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ is compared with the best-adapted homogeneous model, $\mathcal{P}^{\mathsf{DE}}_{\text{gbmm12A}}$, the analysis gives significance only for inputs with a length greater than 130. Therefore, it can be concluded that the stagnation-based dynamic model $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$ effectively outperforms the best-adapted homogeneous model $\mathcal{P}^{\mathsf{DE}}_{\text{gbmm12A}}$.

Table 4.6: Holm test for $\mathcal{P}^{\textsf{DE}}_{\text{gbmm12A}}$ x HePIMs.

Length	**Control**	i	**Algorithm**	p**-value**	α/i
100	$\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$	6	$\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$	1.2258954459292176E-13	0.008
		5	$\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$	1.7734111951013715E-13	0.010
		4	$\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$	1.980967305825471E-8	0.0125
		3	$\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$	2.761583790983588E-7	0.016
		2	$\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$	8.381464733500173E-7	0.025
		1	$\mathcal{P}^{\textsf{DE}}_{\text{gbmm12A}}$	0.7188026245529647	0.05
110	$\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$	6	$\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$	3.6506733156072896E-20	0.008
		5	$\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$	9.571441858241067E-14	0.01
		4	$\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$	9.62783195797665E-13	0.012
		3	$\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$	9.107359384790177E-11	0.016
		2	$\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$	4.996646058399122E-10	0.025
		1	$\mathcal{P}^{\textsf{DE}}_{\text{gbmm12A}}$	0.4515390224098334	0.05
120	$\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$	6	$\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$	1.927919232582314E-21	0.008
		5	$\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$	2.200327289088137E-12	0.01
		4	$\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$	8.169951697851222E-11	0.012
		3	$\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$	1.933815449552944E-10	0.016
		2	$\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$	3.4787444807522945E-8	0.025
		1	$\mathcal{P}^{\textsf{DE}}_{\text{gbmm12A}}$	0.47145170683892584	0.05
130	$\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$	6	$\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$	1.6510072254753912E-27	0.008
		5	$\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$	4.422946724480415E-12	0.01
		4	$\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$	1.2595868719961186E-10	0.012
		3	$\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$	2.9565205977131737E-10	0.016
		2	$\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$	6.824213461036749E-10	0.025
		1	$\mathcal{P}^{\textsf{DE}}_{\text{gbmm12A}}$	0.5891354643621368	0.05
140	$\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$	6	$\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$	1.027115868211875E-31	0.008
		5	$\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$	5.757742850114732E-20	0.01
		4	$\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$	1.5126612148387156E-18	0.012
		3	$\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$	1.749612437192936E-18	0.016
		2	$\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$	2.413141855941992E-16	0.025
		1	$\mathcal{P}^{\textsf{DE}}_{\text{gbmm12A}}$	8.925709019458667E-4	0.05
150	$\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$	6	$\mathcal{P}^{\text{recHetStag}}_{\text{Tr12A}}$	4.288491503871603E-45	0.008
		5	$\mathcal{P}^{\text{Het}}_{\text{Tr12A}}$	3.257446176124881E-31	0.010
		4	$\mathcal{P}^{\text{recHet}}_{\text{Tr12A}}$	9.286667219420533E-24	0.012
		3	$\mathcal{P}^{\text{recHet}}_{\text{gbmm12A}}$	4.629733664111373E-22	0.016
		2	$\mathcal{P}^{\text{Het}}_{\text{gbmm12A}}$	8.052342507094246E-17	0.025
		1	$\mathcal{P}^{\textsf{DE}}_{\text{gbmm12A}}$	2.4159005413133192E-5	0.05

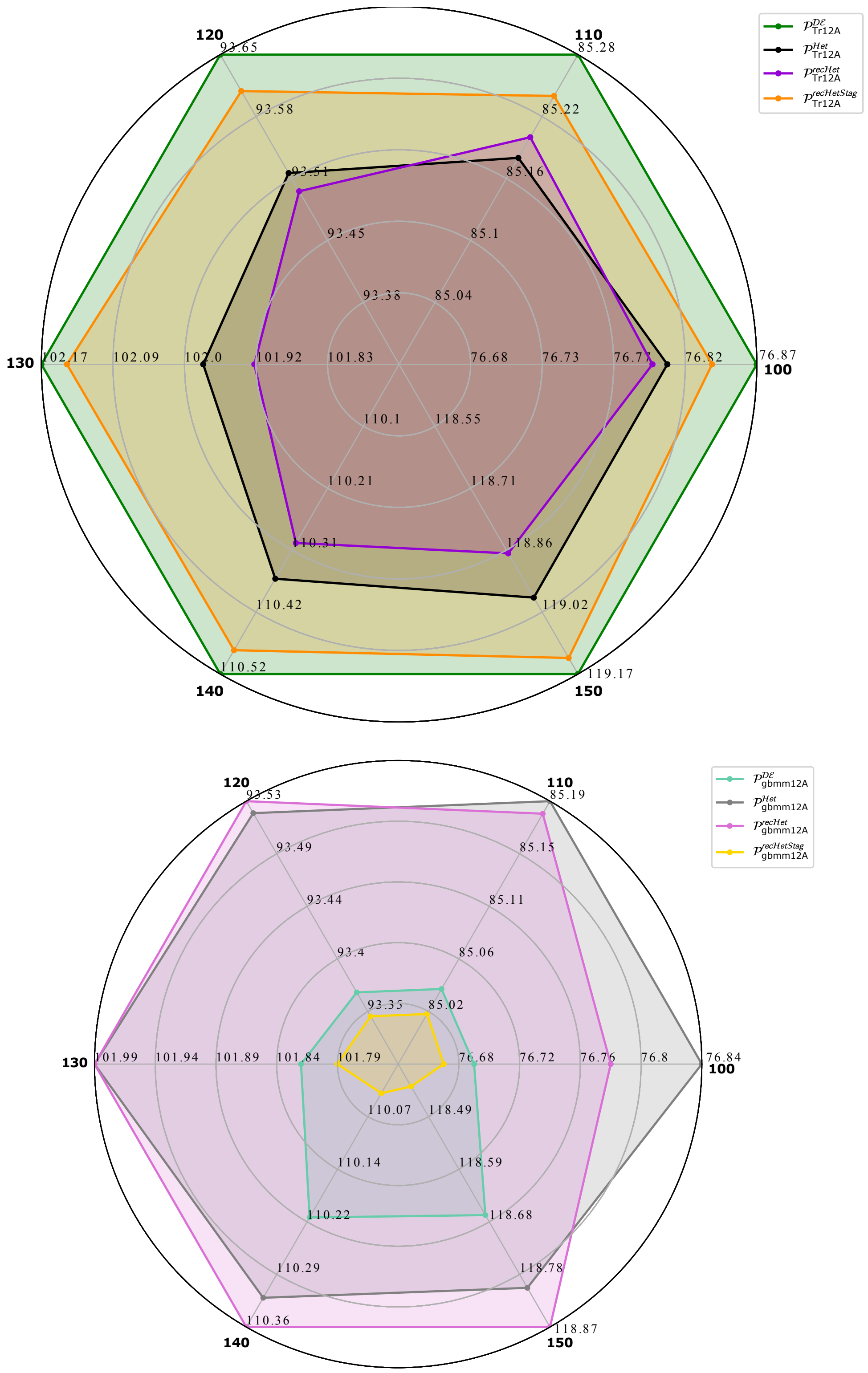


Figure 4.9: Comparing the accuracy of static models: HoPIM from DE and HePIM, and reconfigurable HePIMs, and dynamic models: HoPIM from DE and HePIM, and reconfigurable HePIMs. The charts are scaled according to the model with the worst performance. The HoPIM with static tree topology, $\mathcal{P}^{\text{DE}}_{\text{Tr12A}}$, has the worst accuracy. The dynamic complete graph topology and the stagnation-based reconfigurable HePIM presented the best accuracy.

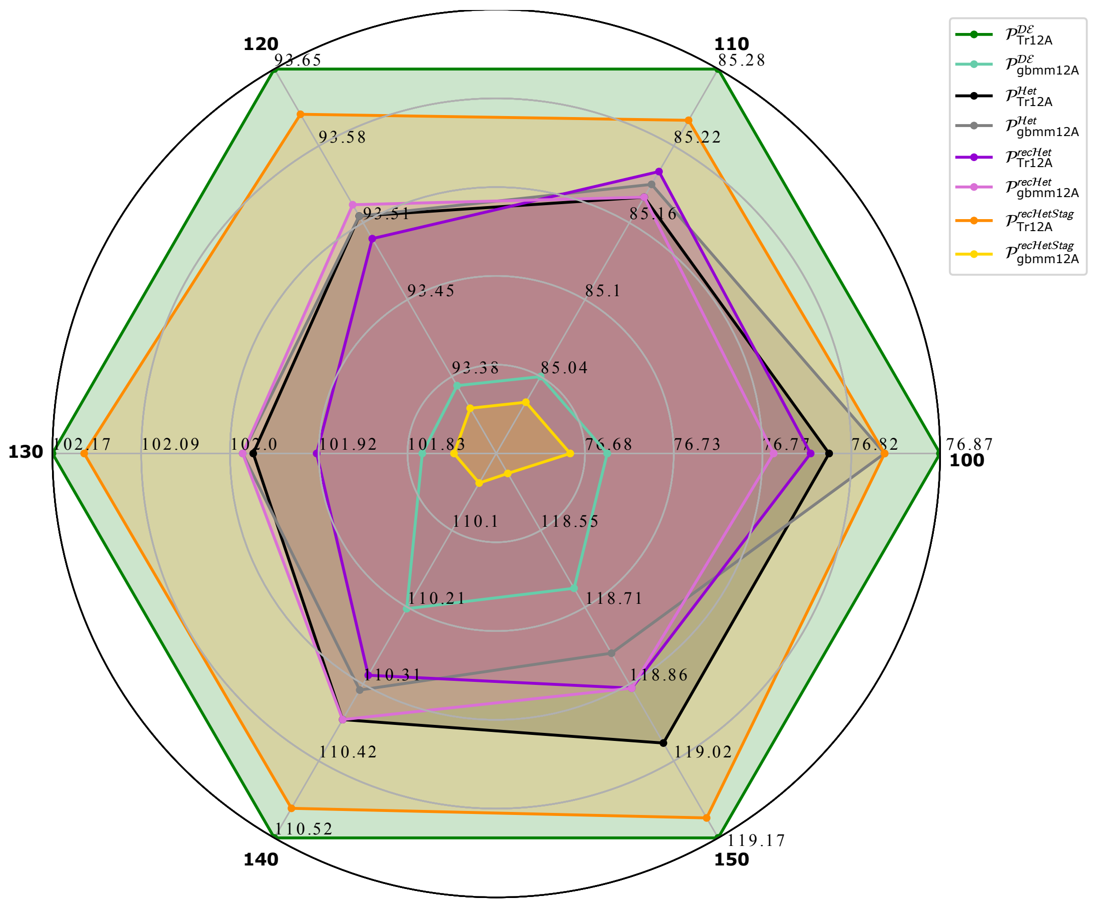


Figure 4.10: Radar chart compiling all accuracies in Figure 4.9. The chart is scaled according to the accuracy of the static homogeneous model $\mathcal{P}^{\text{DE}}_{\text{Tr12A}}$ providing the worst performance. The best accuracy is obtained in all cases by the dynamic stagnation-based reconfigurable heterogeneous model, $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$.

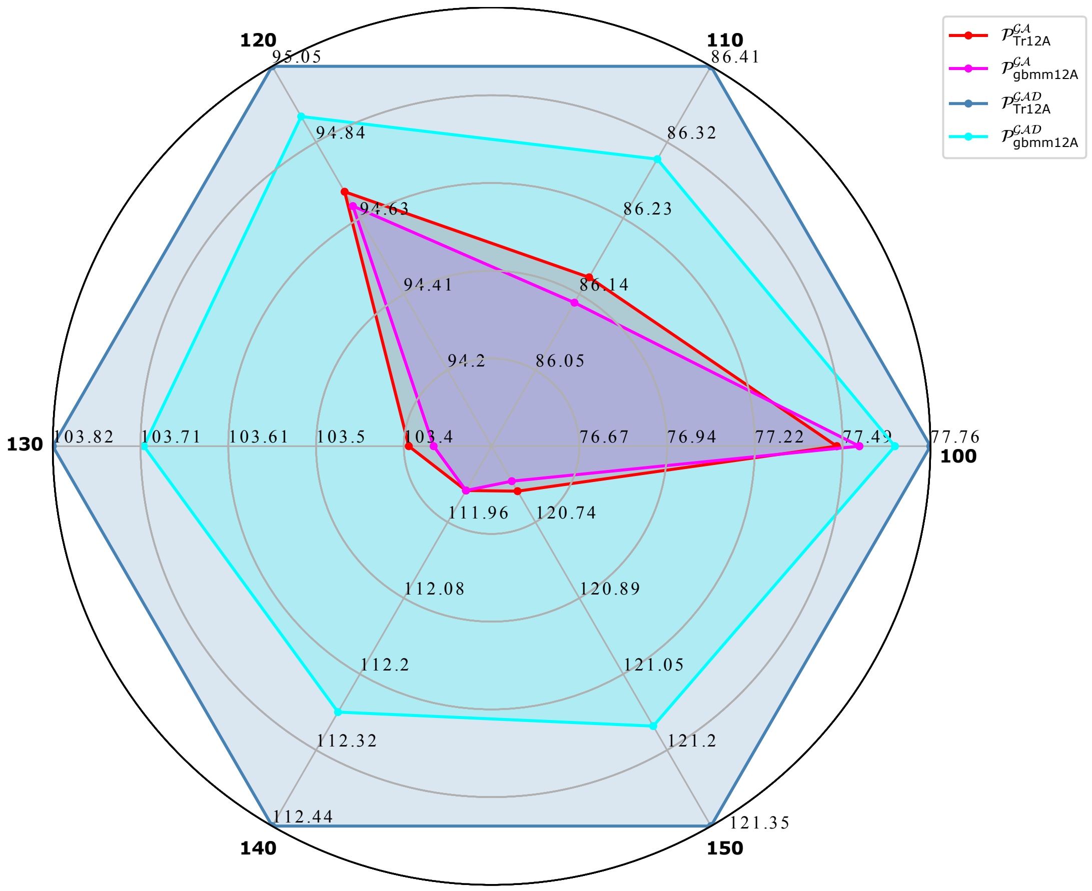


Figure 4.11: The radar chart shows the accuracy obtained by static and dynamic HoPIMs from `GA` and `GAD` with an evolutionary history of 150 generations. Compare this with the radar chart on Figure 4.8 showing that in experiments with 200 generations, not necessarily the `GA` based homogeneous models provide the best performance. Also, in Table 4.4, no reconfigurable model finishes running `GAD` in all its islands.

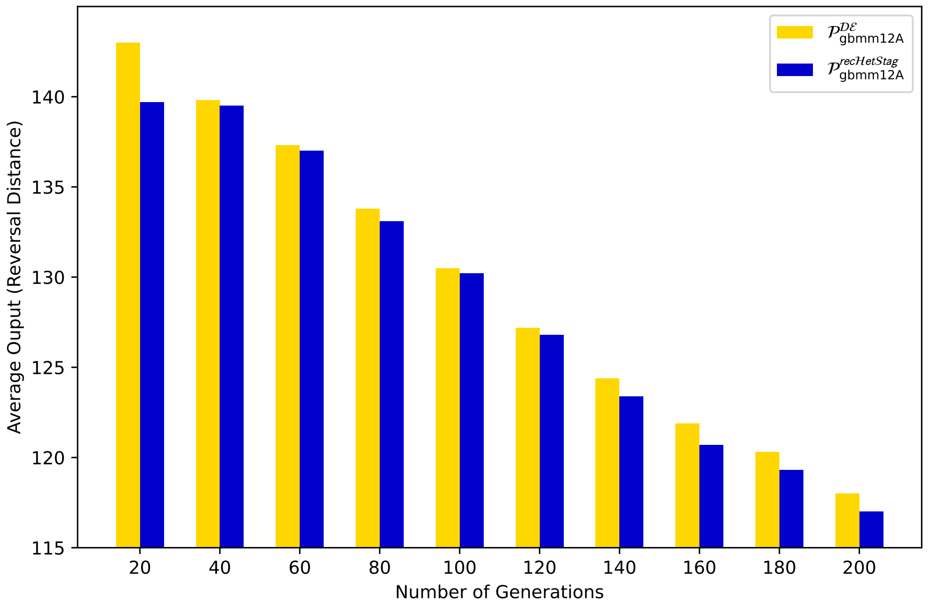


Figure 4.12: Comparing the partial evolution during 200 generations of the best models: HoPIM $\mathcal{P}^{\text{DE}}_{\text{gbmm12A}}$ and reconfigurable HePIM $\mathcal{P}^{\text{recHetStag}}_{\text{gbmm12A}}$.

Chapter 5

Related Work

In this study, well-established migration policies, which play a crucial role in the performance of PIMs, are set through calibrated parameters. Authors such as Grosso [92] and Starkweather *et al.* [93] focused on the migration interval, concluding that high frequency leads to premature convergence, while moderate migration intervals are conducive to discovering more effective solutions. Skolicki and De Jong [94] examined the influence of the migration interval and the number of immigrants and emigrants, determining that the former is a dominant factor. At the same time, the latter plays a secondary role. Additionally, frequent migrations result in islands dominating each other and losing global diversity before exchanging genetic material, a finding also observed in Lissovoi and Witt [38]. Cantú-Paz [9] observed that the response to increasing speed is linked to population convergence caused by selection pressure related to the type of individuals chosen for migration. They noted that convergence occurs quickly with a migration policy where good individuals are selected from local islands to replace the worst individuals on the target island. Conversely, replacing randomly with random individuals has a moderate impact, and convergence occurs slowly when migrants and immigrants are randomly selected. In this work, migration parameters such as the number of migrating individuals, the type of migrating individuals, and the migration interval are calibrated using the tuning method (Eiben and Smit [90]) to maximize accuracy in synchronous and asynchronous PIMs.

We have devised synchronous and asynchronous HoPIMs to explore the superiority of synchronization mechanisms in a general context. Indeed, divergent perspectives exist regarding which alternative yields better results. Early studies suggested that asynchronous HoPIMs could deliver superior speed-ups compared to synchronous architectures (e.g., Alba and Troya [73]). More recently, Abdelhafez *et al.* [11] demonstrated the time superiority of asynchronous architectures, achieving, in some instances, super-linear speed-ups compared to their synchronous counterparts. Nevertheless, various studies have concluded that asynchronous models may offer better accuracy than their synchronous counterparts (e.g., Fernández *et al.* [74], Izzo *et al.* [95]). In contrast to the authors mentioned above, we individually calibrate the parameters of each model to attain the best specific configurations, ensuring a fair and more robust analysis of the overall behavior of each model, be it synchronous or asynchronous.

Some related works have proposed HoPIMs focusing on their adaptability to specific problems. For instance, Federici *et al.* [96] introduced an intriguing class of synchronous PIMs with a unique unidirectional radial dynamic topology, aiming to enhance the quality of solutions for real-world problems involving satellite trajectories. In this study, solution quality was improved using fixed parameter values, with changes made only to the number of individuals and generations. As we aim to provide general insights into the performance of HoPIMs (see [20]), our experiments encompass various calibrated static, dynamic, synchronous, and asynchronous HoPIMs across four $\mathcal{NP}$-Hard problems showed in Chapter 2. Furthermore, experiments use the same resources for all PIMs (number of generations and individuals) to ensure a fair comparison.

Duarte *et al.* [24] proposed a migration policy for PIMs called DIM-1 with target islands defined by attractiveness. The migrations are synchronous to occur *point to point*, where links between islands are unidirectional and weighted dynamically according to the local's attractiveness to the target island. The weights represent the probability of each communication being used for migration. Afterward, they presented a new evaluation strategy in [25] called DIM-2 that changes how to define the island's attractiveness so that islands become more or less attractive according to their solutions' quality. The authors in [97] proposed HePIMs based on strategies DIM-1 and DIM-2 using as an evolutionary engine only variations of DE, which had a successful history in competitions held by the CEC. The results presented demonstrated that the HePIMs from the DIM-1 and DIM-2 can produce better results than the HoPIM versions. Duarte *et al.* also propose tweaks to the migration policy in [26] by adjusting how the attractiveness and weights of island connections are calculated. The adjustments were inspired by the natural phenomenon known as stigmergy proposed by Capriles *et al.* [98]. The dynamic HePIMs highlighted in this study enhance the heterogeneity and dynamism of the migration policy, providing it with algorithmic flexibility achieved through dynamic reconfiguration. Compared with the work of Duarte *et al.*, we note a significant improvement in parameter configuration, carried out through automatic parameter adjustments. In contrast, Duarte performs this task manually and sometimes does not guarantee solutions better than sequential versions (see [25, 26]). Additionally, we employ denser populations and more islands, offering a more robust approach.

Qinxue *et al.* [99] implemented dynamism in PIMs using GA through spectral clustering. The authors do not have a fixed number of islands but limit the number to ten islands. The model is initialized with an island and starts to evolve. During the evolutionary process, whenever there is a migration "epoch" (migration phase), individuals are grouped by similarity using spectral clustering, giving rise to new islands. Similar individuals are assigned to the same island. The standard model is compared with traditional models, and its performance has proved satisfactory. Among the benefits, the authors highlight a reduction of the workload, in contrast to other methods in a parallel environment that usually implement each island allocated to a processor and exchange individuals in the migration phases via messages. The migration policy of this approach is dynamic but does not add the power of algorithmic reconfiguration.

Hashimoto *et al.* [100] proposed a HePIM to solve multi-task problems, where each island evaluates an objective. Migrants are selected at high migration frequency and removed randomly on each local island, replacing the worst individuals in the target islands. Since emigrants went to islands responsible for different objectives, their fitness values are the worst, assuming they have fewer chances of being suitable for the target island objective. The current work applies migration policies shared by Hashimoto *et al.* focusing on consolidating reconfigurability as a new influential parameter to be considered in the design of HePIMs. Despite this restriction, it is clear that the proposed model's heterogeneity is relevant to multi-objective optimization since different BAs adapt better to various optimization tasks.

Lardeux *et al.* [101] study dynamic PIMs with a focus on migration policies at the level of individuals. To control migrations, the authors combine migration and gain matrices that are updated during the search, allowing the simultaneous use of different migration policies within the same model. Thus, individuals cooperate and share their information throughout the evolutionary process. The choices of the best migrations are improved using the QLearning approach, which aims to learn the best migration options by each user. The results show that the QLeaning significantly improves the accuracy of the results. A parallel study addresses the context of population size versus the number of generations, where results show that increasing the number of individuals and reducing the number of generations does not benefit the quality of the results. As in the Qinxue *et al.* clustering-based approach, in contrast to ours, the Lardeux *et al.* method allows variation of island population size through the selection of individual migration choices. Still, it does not apply the flexibility of reconfiguration.

From our side, Silveira *et al.* [14] proposed HePIMs running four different BAs on their islands. The HePIMs maintained population diversity by covering the solution space and reducing overlap between islands compared to HoPIMs. In [15], the authors go a step forward, maintaining the population diversity provided by HePIMs and increasing their flexibility, allowing BA reconfiguration on islands during execution according to their performance, where the islands may substitute their BAs periodically during the evolutionary process. The results were competitive regarding the best-adapted HoPIMs, demonstrating the potential of adding such (periodic) reconfiguration capability to HePIMs. After a series of experiments, the current work (Chapter 4) shows how refining the dynamism of reconfiguration and reaching the stagnation-based reconfiguration approach, such innovative ability adds the required flexibility and diversity to HePIMs to outperform the best-adapted HoPIMs.

Nssibi *et al.* [102] survey nature-inspired metaheuristic methods in the field of machine learning, more specifically for feature selection. In this perspective, the authors include a section highlighting the strength of PIMs in recent works to solve the feature selection problem. The authors discuss works related to HoPIMs, such as the Harris Hawk Optimization algorithm with islands organized over master-slave communication topology. The slave processors have a swarm of hawks, which send the best solutions to the master island when the results of their local optimization are available. Also, they survey models with a moth flame optimization algorithm using random ring topology with an elitist policy: the best immigrants replacing the worst natives. A multi-objective evolutionary procedure with an evolutionary engine based on the non-dominated sorting `GA` is also considered, where the islands are organized through a dynamic complete graph and elitist migration policy. Moreover, they survey parallel `PSO` models where a ring topology organizes the islands, and the migration policy consists of sending the best individual and only replacing the worst native if the immigrant has better fitness. The survey does not address island-based algorithm-adaptative or algorithmic reconfigurable mechanisms like those introduced in the reconfigurable island models mentioned in this paper. Hence, we believe that the innovative reconfigurable PIMs introduced here constitute a significant way for further investigations to improve island-based bio-inspired algorithms.

Another evolutionary mechanism related to reconfiguration is self-adaptation, which is used to update the parameters of an evolutionary procedure dynamically. For instance, in a recent work, Case and Lehre [103] adapted the (μ, λ) algorithm as a discrete and non-elitist evolutionary method incorporating self-adaptation. In the (μ, λ) algorithm, μ

represents the population size, and λ is the mutation parameter responsible for evolving the population. The algorithm incorporates the concept of self-adaptation by dynamically adjusting its parameters during the optimization process in response to changing conditions. Self-adaptation allows the algorithm to adjust the mutation rate dynamically based on information obtained from the ongoing optimization process. Theoretical and experimental analyses suggest that self-adaptation outperforms traditional methods that rely on static parameter values. Although the work does not involve island models, we draw a comparison with our approach, where, instead of applying dynamism to evolutionary parameters, we explore a strategy of algorithm-level dynamism to enhance the global population.

Hyper-heuristics are optimization approaches that seek to automate the local search process, allowing for dynamic and intelligent adaptation. A common strategy is to combine hyper-heuristics with evolutionary algorithms, as discussed in the survey by Dokeroglu *et al.* [104]. This approach typically leverages the principles of evolutionary algorithms by implementing local search methods separately, known as memetic algorithms. Based on the meme's notion, memetic algorithms employ local refinement, perturbation, or constructive methods to enhance solutions. For example, in Silveira *et al.* [105] and Sonco-Alvarez and Ayala-Rincón [106], memetic algorithms were developed to address genome rearrangement problems using a chromosome perturbation heuristic. In another study, Wang and Tang [107] utilized a memetic algorithm supported by machine learning techniques to solve the permutation flow-shop scheduling problem. This method associates solution instances with a file, where all generated solutions are stored for further processing, using machine learning techniques to select high-quality solutions. In another approach, Pereira *et al.* [108] proposed a memetic optimization approach for robotic assembly line balancing, considering equipment scheduling decisions. Wang and Wang [109] proposed a solution for distributed flow-shop scheduling problems: although they do not use an evolutionary algorithm, the proposed method includes simple heuristics employing stochastic local search operations, similar to mutation operators in evolutionary algorithms. Furthermore, some studies have investigated the use of parallelism (see [104]). A specific example involves island models, as proposed by Borgulya [110], where a HoPIM for 2D packing problems is presented. This model employs a memetic algorithm, adopts an approach that learns unfavorable variable values based on the population's worst solutions, and controls each mutation operation's steps.

Chapter 6

Conclusion and Future Work

The research represented a profound and enlightening journey into Parallel Island Models. Throughout this trajectory, we achieved advancements ranging from implementing innovative strategies that underscored the importance of the initial population on the islands to understanding the influence of parameters on result accuracy. We introduced a dynamic genotype-based topology and critically analyzed the efficacy of synchronous and asynchronous models.

While island models have historically shown effectiveness in solving $\mathcal{NP}$-hard problems, our research uncovered the need to delve into frequently concealed details to ensure the robust performance of island models. The migration parameters have emerged as crucial pillars that need calibration to ensure good-quality solutions. We presented evidence that there is no generic configuration capable of guaranteeing competitive solutions for all problems, emphasizing the importance of calibrating parameters according to the specific characteristics of each optimization problem. Additionally, our research underscored the relevance of a careful approach in the initial configuration of homogeneous models, directly impacting the robustness and efficiency of the obtained solutions.

The comparison between synchronous and asynchronous homogeneous models highlighted the specific nuances of each approach, revealing the critical dependence of results on migration parameters and their direct influence on outcomes.

Homogeneous models may offer more effective solutions than heterogeneous models. Such behavior can occur when a homogeneous model demonstrates a remarkable ability to solve a problem effectively. In contrast, the algorithms composing a heterogeneous model may not be equally proficient in solving this specific problem, resulting in inferior accuracy compared to the homogeneous model. In this context, it becomes crucial to employ techniques that can extract and adapt the heterogeneous model to improve its effectiveness.

The introduction of reconfigurable heterogeneous models marked a significant advancement. These models pertinently and dynamically update their local bioinspired algorithms by employing the proposed stagnation-based reconfiguration approach. In contrast to the initially proposed periodic reconfiguration mechanism, where island reconfiguration occurred sparingly according to a fixed generational frequency originally proposed in [15], the stagnation-based reconfiguration mechanism offers the flexibility and variability requirements to surpass all other proposed parallel island models. Statistically evaluated experiments further support the practicality of this reconfiguration technique, demonstrating its efficacy in providing competitive solutions. The results indicate that this approach yields outcomes comparable to reference models that are acknowledged for offering excellent solutions to the unsigned reversal distance problem. This empirical evidence underscores the effectiveness of stagnation-based reconfiguration in enhancing the performance of parallel island models.

From a practical perspective, it can be reasonably inferred that the reconfiguration feature naturally converges to the best-adapted homogeneous model for any optimization problem after a sufficient number of evolutionary generations. However, it is imperative to investigate how dynamically reconfigurable heterogeneity confers advantages at the beginning of evolution, which the model can sustain throughout all generations of the evolutionary process without the necessity of employing the best-adapted bioinspired algorithm. This situation motivates an interest in conducting experiments with alternative bioinspired algorithms.

In the context of other targeted optimization problems, it is crucial to examine how the flexibility of the new reconfigurable mechanism adapts to solve diverse optimization challenges. Notably, algorithmic heterogeneity appears suitable for addressing dynamic multi-objective optimization problems (DMOPs), which involve simultaneously optimizing multiple objectives that change over time. These problems present significant challenges due to the mutable nature of the goals and constraints. The survey by Jiang et al. [111] highlights the complexity of these problems and the effectiveness of population-based algorithms, such as NSGA-II and its dynamic variants, in addressing these challenges. A promising direction for future research is the application of reconfigurable heterogeneous island models in DMOPs. These models would allow for the dynamic replacement of algorithms in islands that are not improving the quality of solutions, promoting greater adaptability and robustness. Developing and testing these approaches across various dynamic benchmarks could significantly advance EDMO research, providing more efficient and resilient solutions in real-world scenarios.

Additionally, considering the success of population size adaptation mechanisms in PSO [112] and DE [113] algorithms, exploring the use of dynamic populations in reconfigurable heterogeneous island models presents a promising avenue for future research. In PSO, the mechanism adapts the population size based on current diversity and deactivates species that are not contributing to solution improvement, allowing for efficient resource allocation. In DE, the diversity adaptation mechanism dynamically adjusts the population size according to measured diversity, increasing or decreasing it to avoid premature convergence and stagnation. Although these strategies are designed for sequential algorithms, they can be easily extended to island models. We plan to implement these dynamic population strategies in our proposed models, enabling continuous adjustments based on the problem's complexity and dynamics.

Another promising direction for future work is the integration of cultural algorithms that utilize normative, historical, situational, and topographical knowledge, as discussed in [114, 115]. This knowledge can be applied to dynamically adjust the algorithms' parameters and reorganize the island model's topology. Normative knowledge can define rules and limits for parameters such as population size and those involved in migrations between islands. Historical knowledge can store and reuse parameter configurations and migration patterns that have been successful in similar conditions. Situational knowledge can adjust the parameters and topology of the islands based on the current state of the populations, such as increasing diversity when necessary or promoting greater exploration. Implementing these approaches can further improve the effectiveness and robustness of the solutions found in our models.

Finally, another possible further direction is to use Lyapunov exponents [116] to enhance the dynamics of island models. Lyapunov exponents are metrics used to quantify the degree of separation of trajectories in dynamic systems. Essentially, they measure sensitivity to initial conditions and can indicate the presence of chaos in the system. A positive Lyapunov exponent suggests chaotic behavior, while a negative exponent indicates stability. These exponents are used to analyze and understand the dynamics of complex systems and predict their behavior over time. Applying such analysis to island models will allow for evaluating population stability in each island, identifying chaotic behaviors, and real-time monitoring of solution quality. Based on Lyapunov metrics, it would be possible to dynamically adjust algorithm parameters, optimize communication topologies between islands, and fine-tune more effective migration and reconfiguration policies.

The proposed island models can be applied to different case studies to evaluate their behavior, being of interest to address problems such as feature selection [117] and automatic hyperparameter tuning [118]. Feature selection involves identifying the most relevant attributes in a dataset to improve the performance of machine learning algorithms. Automatic hyperparameter tuning consists of optimizing the parameters that control the learning process in deep learning models, such as neural networks. Problems that can be addressed include text classification, medical diagnosis, and financial data analysis. In hyperparameter tuning, islands could optimize different sets of hyperparameters for convolutional neural networks, deep regression models, and recurrent neural networks, sharing promising configurations to accelerate convergence and improve performance.

Referências

[1] J. H. Holland. Genetic Algorithms and the Optimal Allocation of Trials. *SIAM Journal on Computing*, 2(2):88–105, 1973. 1, 8

[2] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, pages 39–43, Oct 1995. 1, 8

[3] R. Storn and K. Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997. 1, 9, 25

[4] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *J. of global optimization*, 39(3):459–471, 2007. 1, 9

[5] J. J. Q. Yu and V. O. K. Li. A social spider algorithm for global optimization. *Applied Soft Computing*, 30:614–627, 2015. 1, 9

[6] X. S. Yang and S. Deb. Cuckoo search via lévy flights. In *2009 World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pages 210–214, 2009. 1, 9

[7] S. Mirjalili, S. M. Mirjalili, and A. Lewis. Grey Wolf Optimizer. *Advances in Engineering Software*, 69:46–61, 2014. 1, 9

[8] S. Mirjalili and A. Lewis. The whale optimization algorithm. *Advances in Engineering Software*, 95:51–67, 2016. 1, 9

[9] E. Cantu-Paz. Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms. *Journal of Heuristics*, 7(4):311–334, Jul 2001. 2, 9, 90

[10] D. Sudholt. *Springer Handbook of Computational Intelligence*, chapter Parallel Evolutionary Algorithms, pages 929–959. Springer, 2015. 2, 9

[11] A. Abdelhafez, E. Alba, and G. Luque. Performance analysis of synchronous and asynchronous distributed genetic algorithms on multiprocessors. *Swarm and Evolutionary Computation*, 49:147–157, 2019. 2, 9, 48, 91

[12] L. A. da Silveira, J. L. Soncco-Álvarez, and M. Ayala-Rincón. Parallel memetic genetic algorithms for sorting unsigned genomes by translocations. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 185–192, 2016. 4, 5, 12, 13, 19, 20, 44, 61

[13] L. A. da Silveira, J. L. Soncco-Álvarez, and M. Ayala-Rincón. Parallel genetic algorithms with sharing of individuals for sorting unsigned genomes by reversals. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 741–748, June 2017. 4, 5, 12, 13, 19, 20, 32, 45, 47

[14] L. A. da Silveira, J. L. Soncco-Álvarez, T. A. de Lima, and M. Ayala-Rincón. Heterogeneous Parallel Island Models. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2021. 4, 5, 12, 13, 20, 22, 27, 52, 53, 54, 55, 62, 63, 66, 67, 73, 74, 76, 93

[15] L. A. da Silveira, T. A. de Lima, and M. Ayala-Rincón. Reconfigurable Heterogeneous Parallel Island Models. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1618–1625, 2022. 4, 5, 6, 12, 13, 14, 20, 22, 27, 54, 62, 66, 67, 68, 69, 70, 72, 73, 74, 76, 93, 98

[16] L. A. da Silveira, T. A. de Lima, and M. Ayala-Rincón. On Reconfiguring Heterogeneous Parallel Island Models. *Swarm and Evolutionary Computation*, 2024. 4, 5, 6, 12, 13, 20, 27, 64, 66

[17] L. A. da Silveira, J. L. Soncco-Álvarez, and M. Ayala-Rincón. Parallel Multi-Island Genetic Algotirth for Sorting Unsigned Genomes by Reversals. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2018. 5, 12, 13, 20, 34, 46, 50, 67

[18] L. A. da Silveira, J. L. Soncco-Álvarez, T. A. de Lima, and M. Ayala-Rincón. Parallel island model genetic algorithms applied in np-hard problems. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 3262–3269, 2019. 5, 13, 20, 46

[19] L. A. da Silveira, J. L. Soncco-Álvarez, T. A. de Lima, and M. Ayala-Rincón. Behavior of Bioinspired Algorithms in Parallel Island Models. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020. 5, 13, 20, 22, 25, 48, 66, 67, 69, 73, 76

[20] L. A. da Silveira, T. A. de Lima, J. B. Barros, C. H. Llanos, and M. Ayala-Rincón. On the behavior of parallel island models. *Applied Soft Computing*, 148:110880, 2023. 5, 13, 20, 41, 46, 66, 91

[21] T. Dokeroglu, T. Kucukyilmaz, and E. Talbi. Hyper-heuristics: A survey and taxonomy. *Computers & Industrial Engineering*, 187:109815, 2024. 6, 14

[22] H. Ma, S. Shen, M. Yu, Z. Yang, M. Fei, and H. Zhou. Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey. *Swarm Evol. Comput.*, 44:365–387, 2019. 6, 7, 14, 15

[23] L. A. da Silveira, J. L. Soncco-Álvarez, T. A. de Lima, and M. Ayala-Rincón. Computing translocation distance by a genetic algorithm. In *2015 Latin American Computing Conference (CLEI)*, pages 1–12, 2015. 19, 45

[24] G. Duarte, A. Lemonge, and L. Goliatt. A dynamic migration policy to the island model. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1135–1142, June 2017. 19, 29, 32, 91

[25] G. R. Duarte, A. Lemonge, and L. Goliatt. A New Strategy to Evaluate the Attractiveness in a Dynamic Island Model. In *IEEE Cong. on Evol. Comp. (CEC)*, 2018. 19, 92

[26] G. R. Duarte, A. C. Fonseca, L. G. de Lima, and B. S. L. P. de Lima. An Island Model based on Stigmergy to solve optimization problems. *Natural Computing*, pages 1–29, 2020. 19, 92

[27] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman, 1st edition, 1989. 20

[28] A. Riazi. Genetic algorithm and a double-chromosome implementation to the traveling salesman problem. *SN Applied Sciences*, 1(11):1397, 2019. 20

[29] R. Eberhart and J. Kennedy. Particle swarm optimization. In *IEEE Int. Conf. on neural networks*, volume 4, pages 1942–1948, 1995. 21

[30] K. Yasuda and K. Yazawa. Parameter self-adjusting strategy for particle swarm optimization. In *11th Int. Conf. on Intelligent Systems Design and Applications*, pages 265–270. IEEE, 2011. 22

[31] Y. Mark, Z. Ying, and D. David. Modular robots. *IEEE Spectrum*, 39(2):30–34, 2002. 22

[32] J. J. Q. Yu and V. O. K. Li. A social spider algorithm for global optimization. *Applied Soft Computing*, 30:614–627, 2015. 22

[33] X. Yang and S. Deb. Cuckoo Search via Lévy Flights. In *World Congress on Nature & Biologically Inspired Computing, NaBIC*, pages 210–214. IEEE, 2009. 27, 50

[34] S. Mirjalili, S. M. Mirjalili, and A. Lewis. Grey Wolf Optimizer. *Advances in Engineering Software*, 69:46–61, 2014. 27, 50

[35] S. Mirjalili and A. Lewis. The Whale Optimization Algorithm. *Advances in Engineering Software*, 95:51–67, 2016. 27

[36] A. Askarzadeh. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers & Structures*, 169:1–12, 2016. 27

[37] C. B. Tey, M. R. Leuze, and J. J. Grefenstette. Parallel genetic algorithm. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*, 1987. 29

[38] A. Lissovoi and C. Witt. A runtime analysis of parallel evolutionary algorithms in dynamic optimization. *Algorithmica*, 78(2):641–659, Jun 2017. 29, 90

[39] A. Leitão, F. B. Pereira, and P. Machado. Island models for cluster geometry optimization: how design options impact effectiveness and diversity. *Global Optimization*, 63(4):677–707, 2015. 29, 32

[40] A. Mambrini and D. Sudholt. Design and Analysis of Schemes for Adapting Migration Intervals in Parallel Evolutionary Algorithms. *Evolutionary Computation*, 23(4):559–582, 2015. 29

[41] J. Lässig and D. Sudholt. Design and analysis of migration in parallel evolutionary algorithms. *Soft Computing*, 17(7):1121–1144, Jul 2013. 29, 32

[42] J. Lässig and D. Sudholt. The Benefit of Migration in Parallel Evolutionary Algorithms. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 1105–1112, New York, NY, USA, 2010. ACM. 29

[43] L. Singh and S. Kumar. Migration based parallel differential evolution learning in Asymmetric Subsethood Product Fuzzy Neural Inference System :A simulation study. In *2007 IEEE Congress on Evolutionary Computation*, pages 1608–1613, Sept 2007. 29

[44] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002. 29

[45] E. Cantu-Paz. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 10(2):141–171, 1998. 32

[46] D. Whitley, S. Rana, and R. heckendorn. The island Model Genetic algorithm: On separability, population size and convergence. *CIT. J. of computing and information technology*, 7(1):33–47, 1999. 32

[47] T. C. Belding. The distributed genetic algorithm revisited. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 114–121, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. 32

[48] E. Alba and J. M. Troya. Influence of the migration policy in parallel distributed gas with structured and panmictic populations. *Applied Intelligence*, 12(3):163–181, 2000. 32

[49] T. Hong, W. Lin, S. Liu, and J. Lin. Experimental analysis of dynamic migration intervals on 0/1 knapsack problems. In *2007 IEEE Cong. on Evolutionary Computation (CEC)*, pages 1163–1167, 2007. 32

[50] M. Munetomo, Y. Takai, and Y. Sato. An Efficient Migration Scheme for Subpopulation-Based Asynchronously Parallel Genetic Algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, page 649, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. 34

[51] S. C. Lin, W. F. Punch, and E. D. Goodman. Coarse-grain parallel genetic algorithms: categorization and new approach. In *Proceedings of 1994 6th IEEE Symposium on Parallel and Distributed Processing*, pages 28–37, Oct 1994. 34

[52] V. Bafna and P. A. Pevzner. Genome Rearrangements and Sorting by Reversals. *SIAM Journal on Computing*, 25(2):272–289, feb 1996. 37

[53] D. A. Bader, B. M.E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2004. 38

[54] A. Caprara. Sorting by Reversals is Difficult. In *Proceedings of the First Annual International Conference on Computational Molecular Biology*, pages 75–83, New York, NY, USA, 1997. Association for Computing Machinery. 38

[55] S. Grusea and A. Labarre. The distribution of cycles in breakpoint graphs of signed permutations. *Discrete Applied Mathematics*, 161(10):1448–1466, 2013. 38

[56] T. A. de Lima and M. Ayala-Rincón. On the average number of reversals needed to sort signed permutations. *Discrete Applied Mathematics*, 235(Supplement C):59–80, 2018. 38

[57] J. L. Soncco-Álvarez, D. M. Muñoz, and M. Ayala-Rincón. Opposition-based memetic algorithm and hybrid approach for sorting permutations by reversals. *Evolutionary computation*, 27(2):229–265, 2019. 38

[58] D. Zhu and L. Wang. On the complexity of unsigned translocation distance. *Theoretical computer science*, 352(1):322–328, 2006. 40

[59] S. Hannenhalli. Polynomial-time algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics*, 71(1):137–151, 1996. 41

[60] A. Bergeron, J. Mixtacki, and J. Stoye. On sorting by translocations. *Computational Biology*, 13(2):567–578, 2006. 41

[61] I. P. Gent, C. Jefferson, and P. Nightingale. Complexity of n-Queens Completion. *Journal of Artificial Intelligence Research*, 59:815–848, Aug 2017. 41

[62] H. S. Stone and J. M. Stone. Efficient search techniques-an empirical study of the n-queens problem. *IBM Journal of Research and Development*, 31(4):464–474, July 1987. 41

[63] M. Lazarova. Efficiency of parallel genetic algorithm for solving n-queens problem on multicomputer platform. In *the 9th WSEAS International Conference on Evolutionary Computing*, pages 51–56, 2008. 41

[64] X. Hu, R. C. Eberhart, and Y. Shi. Swarm intelligence for permutation optimization: a case study of n-queens problem. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706)*, pages 243–246, April 2003. 41

[65] W. Wolf, A. A. Jerraya, and G Martin. Multiprocessor System-on-Chip (MPSoC) Technology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(10):1701–1713, oct 2008. 42

[66] G. de Micheli and L. Benini. Networks on chips: 15 years later. *Computer*, 50(5):10–11, May 2017. 42

[67] S. Hesham, J. Rettkowski, and D. Goehringerand M. A. Abd El Ghany. Survey on Real-Time Networks-on-Chip. *IEEE Transactions on Parallel and Distributed Systems*, 28(5):1500–1517, may 2017. 42

[68] L. S. Indrusiak. End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip with priority-preemptive arbitration. *Systems Architecture*, 60(7):553–561, 2014. 42

[69] M. R. Garey and D. S. Johnson. Computers and intractability. a guide to the theory of np-completeness. *Symbolic Logic*, 48(2):498–500, 1983. 42

[70] C. L. Liu and J. W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *ssociation for Computing Machinery*, 20(1):46–61, 1973. 43

[71] J. L. Soncco-Álvarez and M. Ayala-Rincón. A genetic approach with a simple fitness function for sorting unsigned permutations by reversals. In *2012 7th Colombian Computing Congress (CCC)*, pages 1–6, 2012. 45

[72] A.E. Eiben and S.K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011. 45

[73] E. Alba and J. M. Troya. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Future Generation Computer Systems*, 17(4):451–465, 2001. 48, 91

[74] F. Fernández, G. Galeano, and J. A. Gómez. Comparing Synchronous and Asynchronous Parallel and Distributed Genetic Programming Models. In *5th European Conference on Genetic Programming*, pages 326–335. Springer, 2002. 48, 91

[75] Y. Saito, M. Sato, M. Midtlyng, and M. Miyakawa. Parallel and Distributed MOEA/D with Exclusively Evaluated Mating and Migration. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, July 2020. 49

[76] N. Otaka, Y.Fukuyama, Y. Kawamura, K. Murakami, and A. Santana. Refrigerated Showcase Fault Detection by a Pasting based Artificial Neural Networks using Parallel Multi-population Modified Brain Storm Optimization and Correntropy. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, July 2020. 49

[77] W. Liu, P. P. Pokharel, and J. C. Principe. Correntropy: A localized similarity measure. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 4919–4924, 2006. 49

[78] R. Ohira, M. S. Islam, and H. Kayesh. Speedup vs. quality: Asynchronous and cluster-based distributed adaptive genetic algorithms for ordered problems. *Parallel Computing*, 103, 2021. 49

[79] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, pages 39–48, 2000. 50

[80] A. Skakovski and P. Jedrzejowicz. An island-based differential evolution algorithm with the multi-size populations. *Expert Syst. Appl.*, 126:308–320, 2019. 50

[81] A. Skakovski and P. Jedrzejowicz. A multisize no migration island-based differential evolution algorithm with removal of ineffective islands. *IEEE Access*, 10:34539–34549, 2022. 50

[82] B. H. Abed-alguni and M. Barhoush. Distributed grey wolf optimizer for numerical optimization problems. *Jordanian Journal of Computers and Information Technology (JJCIT)*, 04(03):130 – 149, 2018. 50, 51

[83] K. S. S. Ranjini and S. Murugan. Memory based hybrid dragonfly algorithm for numerical optimization problems. *Expert Systems with Applications*, 83:63–78, 2017. 50

[84] C. Yu, L. Kelley, S. Zheng, and Y. Tan. Fireworks algorithm with differential mutation for solving the cec 2014 competition problems. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 3238–3245, 2014. 50

[85] T. Ryoji and A. S. Fukunaga. Improving the search performance of shade using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1658–1665, 2014. 51

[86] Bilal H. Abed-alguni and Noor Aldeen Alawad. Distributed grey wolf optimizer for scheduling of workflow applications in cloud environments. *Applied Soft Computing Journal*, 102(107113), 2021. 51

[87] B. H. Abed-Alguni, A. F. Klaib, and K. M. O. Nahar. Island-based whale optimization algorithm for continuous optimization problems. *International Journal of Reasoning-based Intelligent Systems*, 11(4):319 – 329, 2019. 51

[88] B. H. Abed-alguni and D. Paul. Island-based cuckoo search with elite opposition-based learning and multiple mutation methods for solving optimization problems. *Soft Computing*, 26:3293–3312, 2022. 51

[89] T. Thaher, A. Sheta, M. Awad, and M. Aldasht. Enhanced variants of crow search algorithm boosted with cooperative based island model for global optimization. *Expert Systems with Applications*, 238, 2024. 51

[90] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011. 74, 90

[91] D. G. Pereira, Anabela Afonso, and Fátima Melo Medeiros. Overview of friedman's test and post-hoc analysis. *Communications in Statistics - Simulation and Computation*, 44(10):2636–2653, 2015. 83

[92] P. B. Grosso. *Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model.* PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1985. AAI8520908. 90

[93] T. Starkweather, D. Whitley, and K. Mathias. *Optimization using distributed genetic algorithms*, pages 176–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991. 90

[94] Z. Skolicki and K. De Jong. The influence of migration sizes and intervals on island models. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 1295–1302. Association for Computing Machinery, 2005. 90

[95] D. Izzo, M. Rucinski, and C. Ampatzis. Parallel global optimisation meta-heuristics using an asynchronous island-model. In *2009 IEEE Congress on Evolutionary Computation (CEC)*, pages 2301–2308, 06 2009. 91

[96] L. Federici, B. Benedikter, and A. Zavoli. EOS: a Parallel, Self-Adaptive, Multi-Population Evolutionary Algorithm for Constrained Global Optimization. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–10. IEEE, July 2020. 91

[97] G. R. Duarte and B. S. L. P. de Lima. Differential evolution variants combined in a hybrid dynamic island model. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020. 92

[98] P. Capriles, L. G. Fonseca, H. J. C. Barbosa, and A. C. C. Lemonge. Rank-based ant colony algorithms for truss weight minimization with discrete variables. *Communications in Numerical Methods in Engineering*, 23(6):553–575, 2007. 92

[99] Q. Meng, J. Wu, J. Ellis, and P. J. Kennedy. Dynamic island model based on spectral clustering in genetic algorithm. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1724–1731, 2017. 92

[100] R. Hashimoto, H. Ishibuchi, N. Masuyama, and Y. Nojima. Analysis of Evolutionary Multi-Tasking as an Island Model. In *Proc. of the Genetic and Evol. Comp. Conf. (GECCO)*, pages 1894–1897. ACM, 2018. 92

[101] F. Lardeux, J. Maturana, E. Rodriguez-Tello, and F. Saubion. Migration Policies in Dynamic Island Models. *Natural Computing: An International Journal*, 18(1):163—-179, mar 2019. 93

[102] M. Nssibi, G. Manita, and O. Korbaa. Advances in nature-inspired metaheuristic optimization for feature selection problem: A comprehensive survey. *Computer Science Review*, 49(100559), 2023. 94

[103] B. Case and P. K. Lehre. Self-Adaptation in Nonelitist Evolutionary Algorithms on Discrete Problems With Unknown Structure. *IEEE Transactions on Evolutionary Computation*, 24(4):650–663, 2020. 94

[104] T. Dokeroglu, T. Kucukyilmaz, and E. Talbi. Hyper-heuristics: A survey and taxonomy. *Computers & Industrial Engineering*, 187:109815, 2024. 95, 96

[105] L. A. da Silveira, J. L. Soncco-Álvarez, T. A. de Lima, and M. Ayala-Rincón. Memetic and Opposition-Based Learning Genetic Algorithms for Sorting Unsigned Genomes by Translocations. In *Advances in Nature and Biologically Inspired Computing*, pages 73–85. Springer International Publishing, 2016. 95

[106] J. L. Soncco-Álvarez and M. Ayala-Rincón. Memetic algorithm for sorting unsigned permutations by reversals. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2770–2777, 2014. 95

[107] X. Wang and L. Tang. A machine-learning based memetic algorithm for the multi-objective permutation flowshop scheduling problem. *Computers & Operations Research*, 79:60–77, 2017. 95

[108] J. Pereira, M. Ritt, and Ó. C. Vásquez. A memetic algorithm for the cost-oriented robotic assembly line balancing problem. *Computers & Operations Research*, 99:249–261, 2018. 95

[109] J. Wang and L. Wang. A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling. *Computers & Industrial Engineering*, 168:108126, 2022. 95

[110] I. Borgulya. A parallel hyper-heuristic approach for the two-dimensional rectangular strip-packing problem. *Journal of computing & information technology*, 22(4):251–265, 2014. 96

[111] Jiang, J. Zou, S. Yang, and X. Yao. Evolutionary Dynamic Multi-objective Optimisation: A Survey. *ACM Comput. Surv.*, 55(4), nov 2022. 99

[112] D. Yazdani, D. Yazdani, D. Yazdani, M. N. Omidvar, A. H. Gandomi, and X. Yao. A Species-based Particle Swarm Optimization with Adaptive Population Size and Deactivation of Species for Dynamic Optimization Problems. *ACM Trans. Evol. Learn. Optim.*, 3(4), dec 2023. 99

[113] R. Poláková, J. Tvrdík, and P. Bujok. Differential evolution with adaptive mechanism of population size according to current population diversity. *Swarm and Evolutionary Computation*, 50:100519, 2019. 99

[114] Al. Maheri, S. Jalili, Y. Hosseinzadeh, R. Khani, and M. Miryahyavi. A comprehensive survey on cultural algorithms. *Swarm and Evolutionary Computation*, 62:100846, 2021. 99

[115] R. L. Becerra and C. A. C. Coello. Cultured differential evolution for constrained optimization. *Computer Methods in Applied Mechanics and Engineering*, 195(33):4303–4322, 2006. 99

[116] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano. Determining lyapunov exponents from a time series. *Physica D: nonlinear phenomena*, 16(3):285–317, 1985. 100

[117] N. Pudjihartono, T. Fadason, A. W. Kempa-Liehr, and J. M. O'Sullivan. A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction. *Frontiers in Bioinformatics*, 2, 2022. 100

[118] M. Feurer and F. Hutter. *Hyperparameter Optimization*, pages 3–33. Springer International Publishing, Cham, 2019. 100