Universidade Federal do Paraná

Murilo Zangari de Souza

Innovative hybrid MOEA/D variants for solving multi-objective combinatorial optimization problems

Curitiba PR
2017

Murilo Zangari de Souza

Innovative hybrid MOEA/D variants for solving multi-objective combinatorial optimization problems

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Informática no Programa de Pós-Graduação em Informática, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Aurora Trinidad Ramirez Pozo.

Co-orientador: Roberto Santana.

Curitiba PR
2017

S729i Souza, Murilo Zangari de
Innovative hybrid MOEA/D variants for solving multi-objective combinatorial optimization problems / Murilo Zangari de Souza. – Curitiba, 2016.
136 f. : il. color. ; 30 cm.

Tese - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2016.

Orientador: Aurora Trinidad Ramirez Pozo – Co-orientador: Roberto Santana Hermida.
Bibliografia: p. 103-116.

1. Programação heurística. 2. Otimização combinatória. 3. Combinações (Matemática). 4. Algorítmos. 5. Programação quadrática. I. Universidade Federal do Paraná. II.Pozo, Aurora Trinidad Ramirez. III. Hermida, Roberto Santana . IV. Título.

CDD: 519.7

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
Setor CIÊNCIAS EXATAS
Programa de Pós-Graduação INFORMÁTICA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **MURILO ZANGARI DE SOUZA** intitulada: **Innovative Hybrid MOEA/D variants for solving multi-objective Combinatorial Optimization Problems**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO.

Curitiba, 16 de Dezembro de 2016.

AURORA TRINIDAD RAMIREZ POZO
Presidente da Banca Examinadora (UFPR)

CAROLINA PAULA DE ALMEIDA
Avaliador Externo (UNICENTRO)

ALEXANDER MENDIBURU
Avaliador Externo (UPV)

ROBERTO SANTANA HERMIDA
Avaliador Externo (UPV)

LUIZ CARLOS DE ABREU RODRIGUES
Avaliador Externo (UTFPR)

RICHARD ADERBAL GONÇALVES
Avaliador Externo (UNICENTRO)

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação - Informática

Agradecimentos

Inicialmente, agradeço à minha orientadora, Doutora Aurora Pozo, por ter me concedido esta oportunidade de realizar o doutorado.

Agradeço a oportunidade que me foi dada de realizar o doutorado sanduíche na Universidade do País Basco, com a supervisão dos doutores Roberto Santana e Alexander Mendiburu, os quais acreditaram em mim e contribuíram significativamente para o desenvolvimento desta tese. Certamente foi uma experiência de grande valia para mim.

Agradeço ao Dr. Josu Ceberio, por conceder seus códigos fontes, o qual contribuiu no desenvolvimento deste trabalho.

À CAPES pelo financiamento tanto aqui no Brasil quanto no exterior.

Aos meus colegas de trabalho e laboratório, Olacir Castro Júnior, Giovanni Guizo e Gian Fritsche os quais contribuíram diretamente com discussões e implementações.

Aos meus amigos Diego Catani, Luzia Graciele, Gabriel Goldoni e Tarcísio Bélli por estarem sempre presente nos bons e maus momentos.

À minha irmã Carla Zangari e minha tia Neuza de Souza.

Por fim e não menos importante, agradeço aos meus pais, Edmirce Zangari e Antônio Carlos por todo apoio e compreensão.

Acknowledgement

First of all, I would like to thank my advisor Dr. Aurora Pozo, for the opportunity, her patience, and the partnership during these four years.

I should also like to thank the incredible opportunity given to me to do the internship at the University of the Basque Country, with the advisors Dr. Roberto Santana and Dr. Alexander Mendiburu, who trusted me, and contributed significantly to the development of this doctorate. Certainly, it as one of the most valuable experience in my life.

To Josu Ceberio, for the source codes which were essential for the development.

To CAPES for the funding of this doctorate both in Brazil and abroad.

To my friends of research, Olacir Castro Junior, Giovani Guizo, and Gian Fritsche who directly helped me with advices, implementations, and discussions.

To my friends Diego Catani, Luzia Graciele, Gabriel Goldoni and Tarcisio Bélli for always being by my side in good and bad moments.

To my sister Carla Zangari and my aunt Neuza de Souza.

Finally, but no less important, I thank my parents, Edmirce Zangari and Antônio Carlos de Souza, for the unconditional support and understanding.

Murilo

Resumo

Muitos problemas do mundo real podem ser representados como um problema de otimização combinatória. Muitas vezes, estes problemas são caracterizados pelo grande número de variáveis e pela presença de múltiplos objetivos a serem otimizados ao mesmo tempo. Muitas vezes estes problemas são difíceis de serem resolvidos de forma ótima. Suas resoluções tem sido considerada um desafio nas últimas décadas. Os algoritimos metaheurísticos visam encontrar uma aproximação aceitável do ótimo em um tempo computacional razoável. Os algoritmos metaheurísticos continuam sendo um foco de pesquisa científica, recebendo uma atenção crescente pela comunidade. Uma das têndencias neste cenário é a arbordagem híbrida, na qual diferentes métodos e conceitos são combinados objetivando propor metaheurísticas mais eficientes. Nesta tese, nós propomos algoritmos metaheurísticos híbridos para a solução de problemas combinatoriais multiobjetivo. Os principais ingredientes das nossas propostas são: (i) o algoritmo evolutivo multiobjetivo baseado em decomposição (*MOEA/D framework*), (ii) a otimização por colônias de formigas e (iii) e os algoritmos de estimação de distribuição. Em nossos frameworks, além dos operadores genéticos tradicionais, podemos instanciar diferentes modelos como mecanismo de reprodução dos algoritmos. Além disso, nós introduzimos alguns componentes nos frameworks objetivando balancear a convergência e a diversidade durante a busca. Nossos esforços foram direcionados para a resolução de problemas considerados difíceis na literatura. São eles: a programação quadrática binária sem restrições multiobjetivo, o problema de programação *flow-shop* permutacional multiobjetivo, e também os problemas caracterizados como deceptivos. Por meio de estudos experimentais, mostramos que as abordagens propostas são capazes de superar os resultados do estado-da-arte em grande parte dos casos considerados. Mostramos que as diretrizes do MOEA/D hibridizadas com outras metaheurísticas é uma estratégia promissora para a solução de problemas combinatoriais multiobjetivo.

Palavras-chave: metaheuristicas, otimização multiobjetivo, problemas combinatoriais, MOEA/D, otimização por colônia de formigas, algoritmos de estimação de distribuição, programação quadrática binária sem restrições multiobjetivo, problema de programação *flow-shop* permutacional multiobjetivo, abordagens híbridas.

Abstract

Several real-world problems can be stated as a combinatorial optimization problem. Very often, they are characterized by the large number of variables and the presence of multiple conflicting objectives to be optimized at the same time. These kind of problems are, usually, hard to be solved optimally, and their solutions have been considered a challenge for a long time. Metaheuristic algorithms aim at finding an acceptable approximation to the optimal solution in a reasonable computational time. The research on metaheuristics remains an attractive area and receives growing attention. One of the trends in this scenario are the hybrid approaches, in which different methods and concepts are combined aiming to propose more efficient approaches. In this thesis, we have proposed hybrid metaheuristic algorithms for solving multi-objective combinatorial optimization problems. Our proposals are based on (i) the multi-objective evolutionary algorithm based on decomposition (MOEA/D framework), (ii) the bio-inspired metaheuristic ant colony optimization, and (iii) the probabilistic models from the estimation of distribution algorithms. Our algorithms are considered MOEA/D variants. In our MOEA/D variants, besides the traditional genetic operators, we can instantiate different models as the variation step (reproduction). Moreover, we include some design modifications into the frameworks to control the convergence and the diversity during their search (evolution). We have addressed some important problems from the literature, e.g., the multi-objective unconstrained binary quadratic programming, the multiobjective permutation flowshop scheduling problem, and the problems characterized by deception. As a result, we show that our proposed frameworks are able to solve these problems efficiently by outperforming the state-of-the-art approaches in most of the cases considered. We show that the MOEA/D guidelines hybridized to other metaheuristic components and concepts is a powerful strategy for solving multi-objective combinatorial optimization problems.

Keywords: meta-heuristics, multi-objective optimization, combinatorial problems, MOEA/D, ant colony optimization, estimation of distribution algorithms, unconstrained binary quadratic programming, permutation flowshop scheduling problem, hybrid approaches.

Contents

List of Figures

List of Tables

List of Acronyms

ACO	Ant Colony Optimization
BACO	Binary Ant Colony Optimization
COP	Combinatorial Optimization Problem
DM	Decision Making
EA	Evolutionary Algorithm
EDA	Estimation of Distribution Algorithm
EP	External Pareto
GA	Genetic Algorithm
GM	Graphical Models
GPU	Graphics Processing Unit
GRASP	Greedy randomized adaptive search procedure
HV	Hypervolume
IGD	Inverted Generational Distance
LS	Local Search
MCOP	Multiobjective Combinatorial Optimization Problem
MK	Mallows Kernel
MOEA	Multiobjective Evolutionary Algorithm
MOEA/D	Multiobjective Evolutionary Algorithm based on Decomposition
MOEDA	Multiobjective Estimation of Distribution Algorithm
MOP	Multiobjective Optimization Problem
MoPFSP	Multiobjective Permutation Flowshop Scheduling Problem
mUBQP	multiobjective Unconstrained Binary Quadratic Programming
NSGA	Nondominated Sorting Genetic Algorithm
PBI	Penalty-based Boundary Intersection
PBIL	Population-based incremental learning
PF	Pareto Front
PFSP	Permutation Flowshop Scheduling Problem
PS	Pareto Set
RIPG	Restart Iterated Pareto Greey
SA	Simulated Annealing
SPEA	Strengh Pareto Evolutionary Algorithm
TS	Tabu Search
UBQP	Unconstrained Binary Quadratic Programming
UMDA	Univariate Marginal Distribution Algorithm
VNS	Variable Neighborhood Search

Resumo Estendido

Motivação e Objetivo Geral

Vários problemas do mundo real, tais como, problemas da área de engenharia, ciência da computação, finança e industria podem ser definidos como problemas de otimização combinatória. Frequentemente, estes problemas possuem um grande número de variáveis e a presença de múltiplos objetivos para serem otimizado ao mesmo tempo. Geralmente, esses problemas combinatoriais são difíceis de serem resolvidos de forma ótima, e suas resoluções continuam sendo consideradas um desafio [Miettinen, 2012, Blum e Roli, 2008]. No caso multi-objetivo, a fronteira de *Pareto* (*FP*) real de um problema pode conter um número grande (ou infinito) de soluções (chamadas de soluções ótimas de Pareto). Neste caso, atingir a fronteira real é considerado um problema difícil. Neste caso, os algoritmos de otimização multi-objetivos visam encontrar uma fronteira de Pareto aproximada em um tempo computacional considerado razoável [Miettinen, 2012, Ehrgott e Gandibleux, 2008].

A pesquisa na área de otimização concentra-se no desenvolvimento de algoritmos meta-heurísticos [Glover, 1986, Blum e Roli, 2003, Blum et al., 2011]. O sucesso de meta-heurísticas como: 1) *Arrefecimento Simulado* (AS) [Brooks e Morgan, 1995], 2) Busca Tabu (BT) [Glover, 1989], 3) Algoritmos Genéticos (AG) [Holland, 1975], 4) Otimização por Colônias de Formigas [Dorigo et al., 1996], e 5) Algoritmos de Estimação de Distribuição (AED) [Mühlenbein e Paass, 1996] para resolver problemas de otimização é, hoje, amplamente reconhecido pela comunidade cientifica [Blum et al., 2011]. Estas meta-heurísticas foram logo estendidas para resolver com problemas com dois ou mais objetivos (PMO) [Miettinen, 2012, Coello et al., 2007, Ehrgott e Gandibleux, 2008].

Entretanto, os algoritmos meta-heurísticos propostos podem sofrem de alguma limitação. As principais limitações dos trabalhos da literatura são:

- Os métodos propostos são *orientados ao problema*. Neste caso, os algorítimos são designados para resolver apenas um problema em questão da melhor forma possível, levando em consideração as propriedades intrínsecas do problema. Estes métodos depreciam a generalização do algoritmo em resolver diversos problemas [Blum et al., 2011, Ehrgott e Gandibleux, 2008].

- Uma outra área de pesquisa concentra-se em melhorar a acurácia e/ou a generalização das meta-heurísticas, porém depreciando o custo computacional [Crainic e Toulouse, 2010].

Portanto, o desenvolvimento de meta-heurísticas para resolver problemas de otimização multi-objectivo continua sendo uma área de pesquisa atraente, e continua recebendo atenção [Ehrgott e Gandibleux, 2008, Yenisey e Yagmahan, 2014, Trivedi et al., 2016]. Algumas das linhas de pesquisa atuais em meta-heurísticas podem ser distinguidas como:

1. Abordagem híbridas: Nesta abordagem, diferentes métodos e conceitos são combinados com o intuito de propor novos algoritmos aprimorando a acurácia [Blum et al., 2011, Ehrgott e Gandibleux, 2008].

2. Hiper-heurísticas: Nesta abordagem, o objetivo é propor meta-heurísticas mais gerais, no qual possuem diversos ingredientes/componentes que são configurados automaticamente durante a busca [Burke et al., 2013].

3. Uso de computação paralela: Neste caso, o objetivo é aprimorar a eficiência dos algoritmos meta-heurísticos diminuindo o custo computacional [Crainic e Toulouse, 2010].

Neste tese, nós trabalhamos primeiramentecom as linhas de pesquisa 1 (abordagem híbridas) e 3 (abordagem paralela), resultando em algumas contribuições. Porém, para darmos um foco mais específico para a tese, decidimos trabalhar apenas com a linha de pesquisa 1 (abordagem híbridas).

Assim, podemos dizer que o objetivo geral desta tese é a proposta de meta-heurísticas híbridas para resolver problemas combinatoriais multi-objetivos de forma eficiente. Foram estudados e combinados diversos ingredientes. A seguir, mencionamos os ingredientes chave e a justificativa de suas utilizações.

- **Algoritmos Evolucionários Multi-objetivo (AEMO) [Coello et al., 2007] e a abordagem de decomposição [Miettinen, 2012]:** AEMO são particularmente adequados para resolver PMOs pois eles podem lidar simultaneamente com um conjunto de soluções (chamado de população). Algoritmos baseados em população permitem que diversas soluções não-dominadas sejam encontradas em uma única execução. Eles possuem a habilidade de buscar soluções que compõem diferentes *trade-offs* no espaço objetivo. Além disso, exitem diferentes estratégias para se buscar/manter um conjunto de soluções não-dominadas. Os AEMO baseados em decomposição usam uma função agregação escalar e um conjunto de vetores de pesos, e assim decompõem o problema em questão em um número de subproblemas mono-objectivo. No *framework* MOEA/D (do inglês, *Multi-objective Evolutionary Algorithm based on Decomposition*) [Zhang e Li, 2007], o PMO em questão é decomposto em um conjunto de subproblemas mono-objetivo e eles são otimizados simultaneamente de forma colaborativa usando o conceito de vizinhança entre os subproblemas. Atualmente, as linhas de pesquisa sobre o MOEA/D são varias e incluem: (i) modificações na estrutura do *framework*, (ii) abordagem híbridas, e (ii) aplicação em diversos problemas [Trivedi et al., 2016]. Esses pesquisas motivaram o trabalho desenvolvido nesta tese.

- **Otimização por colônias de formigas:** ACO (do inglês, Ant Colony Optimization) é uma meta-heurística baseada em população e inspirada pelo comunicação indireta de formigas reais por meio de trilhas de uma substância química chamada feromônio [Dorigo et al., 1996]. Computacionalmente, as formigas artificiais são agentes estocásticos que usam a informação que reflete a experiência acumulada pelas formigas anteriores para construir novas soluções. Algorítimos baseados na meta-heurística ACO tem sido aplicados para resolver uma vasta gama de problemas, principalmente os combinatoriais [Dorigo et al., 2006, Dorigo et al., 2008, Mohan e Baskaran, 2012, López-Ibánez e Stutzle, 2012].

- **Algorítimos de Estimação de Distribuição [Mühlenbein e Paass, 1996, Larrañaga e Lozano, 2002]**: são algoritmos evolucionários baseados em população que, a cada geração, aprende um modelo probabilístico a partir de um conjunto de soluções promissoras. Os filhos (novas soluções) são então obtidos executando uma amostragem do modelo probabilístico aprendido. AED foram originalmente propostos para resolver problemas complexos, nos quais os operadores genéticos tradicionais não são muito eficientes em resolver porque eles falham em identificar automaticamente a estrutura do problemas, o qual consiste de várias variáveis correlacionadas. Esta questão é referida na literatura como problemas de aprendizagem ligada.

Na literatura, o *framework* MOEA/D, ACO e AED têm sido aplicados tanto separadamente como também combinados (híbridos) [Ehrgott e Gandibleux, 2008, Ke et al., 2013, Trivedi et al., 2016]. Na literatura há diversos trabalhos mostrando a potencialidade destes métodos [Yenisey e Yagmahan, 2014, Trivedi et al., 2016].

Objetivos Específicos e Contribuições

As meta-heurísticas híbridas propostas nesta tese são consideradas variações do framework MOEA/D pois elas seguem as suas principais premissas. Em nosso *framework* diferente modelos (meta-heurísticas) podem ser aplicadas como mecanismo de reprodução, considerando o problema em mãos. Nós distinguimos nossa proposta em três *frameworks* diferentes no qual depende do tipo de modelo combinado com o MOEA/D. A seguir, descrevemos os três *frameworks* e o tipo de problema combinatorial no qual cada um é designado a resolver.

1. MOEA/D combinado com a meta-heurística ACO Binário (MOEA/D-BACO *framework*) aplicado para resolver problemas combinatoriais multi-objectivo com representação binária.

2. MOEA/D hibridizado com modelos gráficos probabilísticos (MOEA/D-GM *framework*) (do inglês, *Graphical Models*) proposto para resolver problemas combinatoriais deceptivos multi-objectivo;

3. MOEA/D hibridizado com o AED basedo em modelos *Mallows* (MEDA/D-MK *framework*) (do inglês, *Multi-objective Estimation of Distribution Algorithm based on Decomposition and kernel of Mallows model* suscetível para resolver problemas combinatoriais multi-objectivo baseados em permutação.

Para mostrar a viabilidade destes *frameworks* propostos, diversos estudos experimentais foram conduzidos. Os frameworks foram aplicados para os seguintes problemas: (i) A programação quadrática binária sem restrições multi-objetivo, do inglês *multi-objective unconstrained quadratic programming* (mUBQP) [Liefooghe et al., 2014], o qual pode representar diversos problemas binários. (ii) Funções/instancias deceptivas [Pelikan et al., 2007], que representam uma outra classe de problemas complexos. (iii) E também o problema da programação *flow-shop* permutacional multi-objectivo (do inglês, *multi-objective permutation flowshop scheduling problem* (MoPFSP) [Minella et al., 2008], no qual é um dos problemas de permutação mais estudados devido a sua complexidade e seus campos de aplicação.

Nosso trabalho resultou em diferentes contribuições, e assim obtivemos publicações em conferencias de grande prestigio e também em revistas cientificas relevantes. A seguir, descrevemos as contribuições:

1. A primeira contribuição é relacionado à abordagem paralela. A investigação foi iniciada com a meta-heurística ACO devido a sua vasta utilização para resolução de problemas combinatoriais. Em geral, algoritmos baseados em ACO são mais custosos que outras meta-heurísticas, tal como os algorítimos genéticos. Por outro lado, algoritmos baseados em ACO são suscetíveis à paralelização [DeléVacq et al., 2013]. Portanto, nós propusemos uma implementação em GPU (do inglês, Graphics Processing Uniti) do algoritmo MOEA/D combinado com ACO (chamado de MOEA/D-ACO) [Ke et al., 2013] usando NVIDIA CUDA[1] com o objetivo de aperfeiçoar seu tempo computacional. Nós reportamos *speedups* acima de 19x para o problema do caixeiro viajante multi-objetivo, e 11x para o problema da mochila multi-objetivo. Entretanto, nós decidimos mudar o foco da nossa investigação

[1] Guia de programação CUDA V5.5. Disponível em http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf

devido as limitações encontradas nesta linha de pesquisa, como hardware disponível e falta de experiência com paralelismo. Logo, nós voltamos nossas atenções apenas para as abordagem híbridas com o intuito de atingir contribuições mais significativas. Nós apresentamos o trabalho desenvolvido com a abordagem paralela no Apêndice B desta tese.

2. Nós propusemos uma nova abordagem para resolver o problema mUBQP [Liefooghe et al., 2014] pois poucas meta-heurísticas foram testadas para resolver este problema. Além disso, o tradicional MOEA/D-ACO não é adequado para resolver instancias do mUBQP com um grande número de variáveis porque a complexidade dos grafos $n \times n$ aumentam exponencialmente com o tamanho do problema. Portanto, nós apresentamos um algoritmo híbrido baseado no MOEA/D e na versão binária do ACO, chamado *Binary ACO (ACO)* [Fernandes et al., 2007]. Algoritmos baseados no BACO agem construindo trilhas de feromônios sobre um grafo $2 \times n$ que representa pseudo-soluções. A principal diferença entre *ACO* e *BACO* é a estratégia usada para representar e prover o espaço de busca para explora-lo. Nós também incluímos alguns ingredientes no MOEA/D-BACO afim de de melhorar sua acurácia. Os resultados mostram que o MOEA/D-BACO supera significativamente o MOEA/D (que aplica operadores genéticos) na maioria das instâncias consideradas. Além do mais, o algoritmo produz resultados competitivos comparado as melhores fronteiras de *Pareto* presentes na literatura para o *benchmark* considerado.

3. Por conseguente, nós seguimos com a nossa pesquisa em novas abordagem para resolver problemas combinatoriais multi-objetivo. Assim, começamos a investigar os algoritmos de distribuição de probabilidades. Primeiro, estudamos os AED que não consideram dependência entre as variáveis por serem mais simples e conter similaridades com o ACO. O AED chamado PBIL (do inglês, *population-based incremental algorithm*) [Mühlenbein e Paass, 1996] é um dos primeiros AED propostos, e ele tem sido aplicado para resolver diversos problemas. Nós mostramos que as diferentes aplicações do PBIL reportadas na literatura correspondem, de fato, em dois algoritmos distinto pois o mecanismo de aprendizado implementado é diferente. Nós analiticamente e empiricamente mostramos que o mecanismo de aprendizagem afeta diretamente no comportamento do algoritmo em termos de convergência e diversidade. Como conclusão, mostramos que este fator deve ser levado em consideração ao aplicar um algorítimo baseado no PBIL para resolver um determinado problema.

4. Indo mais afundo na questão "*O que faz uma instância de um problema combinatorial difícil?*", nós investigamos as propriedades das instâncias do mUBQP. O nível de dificuldade das instâncias do mUBQP propostas por [Liefooghe et al., 2014] dependem do número de variáveis, do arranjo de valores que compõem as matrizes, e a correlação entre os objetivos. Nós seguimos um caminho diferente para gerar novos casos do problema. Em nossa abordagem, o nível de dificuldade é dado pelo número de variáveis co-relacionadas, no qual pequenos blocos de funções deceptivas são plantadas nas matrizes que definem o problema. Nós mostramos que as instâncias introduzidas são mais difíceis de serem resolvidas do que as que são geradas randomicamente.

5. Seguindo com a pesquisa em AED, nos concentramos na exploração de modelos mais complexos devido a sua viabilidade para resolver problemas que possuem dependência entre as variáveis [Pelikan et al., 2007, Mendiburu et al., 2012, Larrañaga et al., 2012]. Neste caso, a distribuição de probabilidades é representada via modelo gráfico, no qual probabilidades condicionais são aprendidas Portanto, nós investigamos a usabilidade dos modelos gráficos combinados com o MOEA/D. O *framework* proposto chamado

MOEA/D-GM pode aprender tanto modelo uni-variados quanto multi-variados como mecanismo de reprodução de cada subproblema. Para avaliar a abordagem proposta, nós utilizamos a função multi-objectivo *Trap5* e as instâncias do mUBQP. Os resultados mostram que a instanciação chamada MOEA/D-Tree, no qual modelos baseados em árvore são aprendidos a partir das matrizes de informação condicional, é capaz de capturar a estrutura da função *Trap5*. MOEA/D-Tree supera de forma significativa o MOEA/D que usa (i) operadores genéticos e (ii) modelos uni-variados. Entretanto, para as instâncias mUBQP, o MOEA/D-Tree foi superado pelos modelos mais simples.

6. Por fim, mas não menos importante, nós investigamos o uso de AED para resolver problemas multi-objetivo baseados em permutação. Neste caso, exploramos os modelos exponenciais baseados em distância, tais como modelo Mallows (MM) e modelo Mallows Generalizado (GMM) [Ceberio et al., 2014a], pois eles têm demonstrado sua relevância no contexto de AED para resolver problemas de permutação mono-objetivo. Entretanto, estes modelos não haviam sido testados no caso multi-objetivo. Portanto, nós introduzimos um novo *framework* baseado em decomposição e no uso de *kernel of Mallows models*, chamado MEDA/D-MK. Para demonstrar a validade da abordagem, executamos diversos experimentos utilizando o problema MoPFSP minimizando diferente combinações de objetivos. Os resultados mostram que o MEDA/D-MK supera a versão aperfeiçoada do MOEA/D especialmente adaptada para o problema em questão. Além disso, nosso *framework* atinge resultados melhores que as melhores fronteiras aproximadas de *Pareto* conhecidas reportadas na literatura para o *benchmark* considerado.

Em geral, por meio de estudos experimentais, mostramos que as abordagem propostas são capazes de superar os resultados do estado-da-arte em grande parte dos casos considerados. Como conclusão geral, dizemos que o MOEA/D hibridizado com outras meta-heurísticas é uma estratégia promissora para a solução de problemas combinatoriais multi-objetivo.

Chapter 1

Introduction

1.1 Motivation and general goal

Several real-world problems from different fields, such as engineering, computer science, finance, and industry, can be stated as combinatorial optimization problems (COPs). Very often, these problems are characterized by the large number of variables and the presence of multiple conflicting objectives. Usually, COPs are hard to solve optimally, and their solutions have been considered a research challenge for a long time [Miettinen, 2012, Blum e Roli, 2008]. In the multi-objective case, the *true Pareto front* (*PF*) of a problem may contain a large (or an infinite) number of Pareto-optimal solutions. Achieving the exact *true PF* from a complex multi-objective optimization problem can be quite difficult. The goal of a multi-objective optimization algorithm is to find an acceptable *PF* approximation in a reasonable computational time [Miettinen, 2012, Ehrgott e Gandibleux, 2008].

In the last three decades, much of the research effort in the optimization area have focused on developing *approximation solution methods* so called metaheuristics [Glover, 1986, Blum e Roli, 2003, Blum et al., 2011]. The success of metaheuristics, such as Simulated Annealing (SA) [Brooks e Morgan, 1995], Tabu Search (TS) [Glover, 1989], Genetic Algorithms (GA) [Holland, 1975], Ant Colony Optimization (ACO) [Dorigo et al., 1996, Dorigo e Gambardella, 1997], Estimation of Distribution Algorithms (EDAs) [Mühlenbein e Paass, 1996, Larrañaga e Lozano, 2002] for solving single-objective combinatorial optimization problems is well recognized today by the research community [Blum et al., 2011]. These solution methods were soon extended to deal with multi-objective COPs (MCOPs) [Miettinen, 2012, Coello et al., 2007, Ehrgott e Gandibleux, 2008].

However, in most of the cases, those proposed methods suffer from one of these following limitations:

- The proposed methods are *problem-oriented*. They were intended to solve the problem at hand in the best way possible taking into account the intrinsic properties of the MCOP at hand but depreciating the generalization of the method to solve different problems [Blum et al., 2011, Ehrgott e Gandibleux, 2008];
- Some works were designed to increase the accuracy and the generalization of the metaheuristic algorithms but depreciating the computational cost [Crainic e Toulouse, 2010].

Therefore, the development of efficient metaheuristic algorithms for solving MCOPs remains an attractive research area and receives growing attention [Ehrgott e Gandibleux, 2008, Yenisey e Yagmahan, 2014, Trivedi et al., 2016]. The current research lines on metaheuristics can be distinguished as follows:

1. Hybrid approaches [Ehrgott e Gandibleux, 2008, Blum et al., 2011] to take advantage of the different methods and concepts combined to improve their accuracy;

2. More general (high-level) metaheuristics (*hyper-heuristics*) [Burke et al., 2013] to improve their generality to self-adapt and solve different problems;

3. Parallel methods [Crainic e Toulouse, 2010] aiming to improve their computational cost.

So, we had several research directions in which we could have given focus. We have started working mutually with item 1 (hybrid approaches) and item 3 (parallel methods), and we have achieved some contributions. However, to have a more specific focus, we have decided to work deeply only on the first research line (item 1).

Therefore, the general goal of this thesis is to propose new (hybrid) metaheuristic algorithms to solve MCOPs efficiently. We have studied and combined several ingredients. Next, we describe these key ingredients, justifying the reason why they are useful to achieve our goal.

- **Multi-objective Evolutionary Algorithms (MOEAs) [Coello et al., 2007] and the decomposition approach [Miettinen, 2012]:** MOEAs are particularly suitable to solve MOPs because they deal simultaneously with a set of possible solutions (population). This concept allows finding several members of the *Pareto* optimal set (PS) in a single "run" of the algorithm. They have the ability to search partially ordered spaces for several *trade-offs*. Moreover, there exist different strategies in which a MOEA can maintain a set of non-dominated solutions. The decomposition-based MOEAs use a scalar aggregation function with a set of weight vectors. Specially, in the MOEA/D framework [Zhang e Li, 2007] a MOP is decomposed into a number of single-objective scalar optimization subproblems and they are optimized simultaneously in a collaborative manner using the neighborhood concept. Current lines of research on MOEA/D are various, such as: 1) design modifications, 2) hybridization with other metaheuristics, and 3) the application to different problems [Trivedi et al., 2016]. These trends on MOEA/D have motivated the work developed in this thesis.

- **Ant Colony Optimization:** ACO is a population-based metaheuristic inspired by the indirect communication of real ants through trails of a chemical substance called pheromone [Dorigo et al., 1996]. Computationally, the *artificial ants* are stochastic agents that use the information that reflects the experience accumulated by the previous artificial ants to construct new solutions. Furthermore, ACO allows the incorporation of underlying heuristics to guide the search. ACO have been applied to solve a broad range of combinatorial optimization problems [Dorigo et al., 2006, Dorigo et al., 2008, Mohan e Baskaran, 2012, López-Ibánez e Stutzle, 2012].

- **Estimation of Distribution Algorithms:** EDAs [Mühlenbein e Paass, 1996, Larrañaga e Lozano, 2002] are a type of population-based evolutionary algorithm that, at each generation, learn from a probabilistic model a set of promising solutions. The new offspring is then obtained by sampling the probabilistic model learned. EDAs were originally proposed to solve some complex problems, in which the traditional genetic operators are not very successful because they fail in automatically identifying the structure of the problem, which consists of several correlated variables. This issue is referred in the literature as the *linkage-learning* problem.

Over time, MOEA/D framework, ACO and EDAs have been studied separately as well as combined [Ehrgott e Gandibleux, 2008, Ke et al., 2013, Trivedi et al., 2016]. The literature

have shown the potentiality of these methods for solving a wide range of multi-objective problems [Yenisey e Yagmahan, 2014, Trivedi et al., 2016].

1.2 Specific goals and contributions

Our approaches are considered MOEA/D variants because they follow the main MOEA/D guidelines. They are instantiated from a unified framework, where different models (metaheuristics) can be applied as the reproduction operator according to the problem representation at hand. We distinguish our proposals in three different frameworks depending on the kind of model combined to MOEA/D. The frameworks and their respective kind of problems addressed are as follows:

1. MOEA/D combined to Binary ACO (MOEA/D-BACO framework) to solve any MCOP with binary representation;

2. MOEA/D using Graphical Models EDA (MOEA/D-GM framework) attempting to solve complex MCOPs with deception;

3. MOEA/D hybridized to Mallows Kernel EDA (MEDA/D-MK framework) to solve permutation-based MCOPs.

We have essentially addressed some representative hard problems from the literature which remain a research trend: (i) the multi-objective unconstrained binary quadratic programming (mUBQP) [Liefooghe et al., 2014], which is able to represent various problems with binary representation and large-scale, (ii) the multi-objective permutation flowshop scheduling problem (MoPFSP) [Minella et al., 2008], which is one of the most studied problems of this kind due to its fields of application, and (iii) the multi-objective deceptive *Trap* function [Pelikan et al., 2007]. These problems have their particular representations and challenges that have to be faced.

Our work has resulted in different contributions, and they have been published in prestigious annual conferences. They have also been accepted for publication in relevant scientific journals. The complete list of papers is presented in Chapter 8 (Final Considerations). In the following, we summarize the contributions achieved for this thesis.

1. The first contribution is related to the parallel approach. We have initially worked with the ACO metaheuristic. In general, ACO algorithms are more expensive than other approaches such as GA. On the other hand, ACO algorithms are suitable to parallelization [DeléVacq et al., 2013]. Therefore, we have proposed a Graphics Processing Unit (GPU) implementation of the MOEA/D hybridized with ACO (MOEA/D-ACO) [Ke et al., 2013] using NVIDIA CUDA[1] in order to improve its computational cost. We have reported speedups up to 19x for the multi-objective traveling salesman problem (MoTPS), and 11x for multi-objective knapsack problem (MOKP). However, we have changed the focus of our research due to some limitations regarding the possibility to achieve more valuable contributions. We refer this subject in Appendix B since it was not the main scope of this thesis.

2. The second contribution is related to the mUBQP problem. Few metaheuristics have been studied to solve it [Liefooghe et al., 2014]. The conventional MOEA/D-ACO is not suitable to solve large-size mUBQP instances because of the complexity to maintain several

[1]CUDA C Programing Guide v5.5. http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf

pheromone matrices ($n \times n$) that grows exponentially with the problem size. Therefore, we have proposed a hybrid algorithm based on MOEA/D framework and the Binary ACO metaheuristic [Fernandes et al., 2007]. BACO algorithms act by building pheromone maps over a $2 \times n$ graph of possible trails representing pseudo-solutions. The main difference between ACO and BACO is the strategy used to represent and provide the search space in order to explore/exploit it. We have also included some features to the MOEA/D-BACO. The components (i) mutation-like effect, (ii) diversity preserving scheme, and (iii) varied neighborhood size were incorporated into the framework to enhance its search ability. We show that the MOEA/D-BACO significantly outperforms MOEA/D in most of the test instances. Moreover, the algorithm has produced competitive results in comparison to the *best-known* approximated *PFs* from the literature for the benchmark considered.

3. Next, we have focused on investigating other metaheuristics to solve MCOPs. First, we have studied EDAs that consider no dependence between the variables because of their simplicity and their similarity to GA and ACO. The population-based incremental algorithm (PBIL) [Mühlenbein e Paass, 1996] is one of the first algorithms of this kind, and it has been extensively applied to many optimization problems. We have shown that the different applications of PBIL reported in the literature correspond, in fact, to two essentially different algorithms, which are defined by the way the learning step is implemented. We have analytically and empirically studied the impact of the learning method on the search behavior. As a result, we show examples in which the choice of a PBIL variant can produce qualitatively different outcomes regarding the search process.

4. Going deep into the question "*what makes a MCOP instance difficult?*", we have investigated the properties of the mUBQP problem. The level of difficult of the mUBQP instances proposed by [Liefooghe et al., 2014] depends on the number of variables, the range of values of the matrix cells, and the correlation strength between the objectives. We have followed a different path to generate cases of the problem. In our strategy, the level of difficulty is given by the number of correlated variables. We present a parametrical approach in which small building blocks of deceptive functions are planted into the matrices that define the mUBQP. Further, we describe the algorithm for creating the new instances. Our experimental results confirm that the introduced instances are indeed harder for our MOEA/D than those that are randomly generated. This study also throw light on a number of issues that influence the complexity of mUBQP instances and that should be taken into account at the time of using this problem as a MOEA benchmark.

5. Following with our research on EDAs, we have directed efforts to explore more complex EDAs due to their useful application for solving hard COPs [Pelikan et al., 2007, Mendiburu et al., 2012, Larrañaga et al., 2012]. These EDAs attempt to learn the statistical dependencies between the variables trying to represent a more accurate model. In this case, the probability distribution is represented by a graphical model. Therefore, we have investigated probabilistic graphical models (PGM) within MOEA/D. The proposed MOEA/D-GM is able to instantiate both univariate and multivariate models for each scalar optimization subproblem. To validate the introduced framework, we have evaluated it to solve a multi-objective version of the deceptive function *Trap5* and the mUBQP instances. The results show that a variant of the framework (so called MOEA/D-Tree), in which tree models are learned from the matrices of the mutual information between the variables, is able to capture the structure of the *bi-Trap5*. Also, the results indicate that MOEA/D-Tree significantly outperforms MOEA/D using (i) genetic operators and (ii) univariate models.

However, MOEA/D-Tree is outperformed by the other algorithms regarding the mUBQP instances.

6. Finally, and no less important, we have investigated EDAs for MCOPs represented by a permutation. In this case, the distance-based exponential models, such as Mallows Model (MM) and Generalized Mallows Model (GMM) [Ceberio et al., 2014a], have demonstrated their validity in the context of EDAs to deal with single-objective permutation-based optimization problems. However, they have not been tested for MCOPs. Therefore, we have introduced a novel general multi-objective decomposition-based EDA using kernels of Mallows Models (MEDA/D-MK framework) for solving multi-objective permutation-based optimization problems. In order to demonstrate the validity of the MEDA/D-MK, we have applied it to solve the MoPFSP minimizing different combination of objectives. The results show that MEDA/D-MK outperforms an improved MOEA/D variant specific tailored for minimizing makespan and total flowtime. Further, our approach achieves better results than the *best-known* approximated *PFs* reported in the literature for the benchmark considered.

1.3 Methodology

The scientific methodology that we have used to achieve the contributions is as follows:

1. We have conducted a systematic review on each target MCOP, indicating their state-of-the-art approaches, and their challenges;

2. We have developed new approaches to solve each MCOP at hand, based on the hybridization of the different components and strategies;

3. In order to validate the proposal, we have conducted several experimental studies on benchmark test instances considering a number of algorithmic aspects, such as: 1) parameter settings, 2)performance assessment, and 3) statistical tests.

4. Finally, we have described the results, and thus establishing the conclusions.

1.4 Organization

The remaining of the thesis is organized as follows:

- Chapter 2 serves as the background, where we present an overview of 1) metaheuristics, 2) MOEA/D framework, and 3) the MCOPs.

- Chapter 3 presents the MOEA/D-BACO applied to solve the mUBQP.

- Chapter 4 presents the analysis of the different learning mechanisms of the PBIL variants.

- Chapter 5 involves the study on the design of hard mUBQP instances.

- Chapter 6 presents the MOEA/D-GM, a MOEA/D framework with Graphical Models.

- Chapter 7 address the MEDA/D-MK framework, and the case of study MoPFSP.

- Chapter 8 draws the final considerations, and future work.

Chapter 2

Background

In this chapter, we first present an overview of optimization and metaheuristics. Next, the concept of single-objective optimization is extended to the nature of multi-objective optimization and the Pareto optimality theory. Furthermore, we describe the MOEA/D framework, the mathematical formulation of the MCOPs addressed, and the experimental methodology used to evaluate the proposed algorithms.

2.1 Optimization and Metaheuristics

The goal in optimization problems is to find a combination of values for a finite set of problem variables, in which these values satisfy constraints, and specify the optimal *solution(s)* with respect to some function(s) that minimize or maximize some objective(s).

Mathematically, we define a solution as a vector $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ where n is the number of problem variables[1]. The *search space* (also called *decision space*), represented as Ω, defines all possible feasible values of $\mathbf{x}$ from some universe. If the *search space* is represented by real numbers, the problem is called *continuous*, if the search space only takes a finite set of distinct values, the problem is called *combinatorial*.

$f(\mathbf{x})$ is a evaluation function (also called *objective function* or *criterion*) to be optimized. The optimization can be a maximization or a minimization of $f(\mathbf{x})$. In the single-objective case, the *objective space*, defined as R, is composed by all the possible values of $f(\mathbf{x})|\mathbf{x} \in \Omega$ and satisfies all the problem constraints. Equation 2.1 formulates the single-objective optimization problem.

$$\begin{aligned} &maximize \ \ (or \ \ minimize) \ \ f(\mathbf{x}) \\ &subject \ \ to \ \ \mathbf{x} \in \Omega \ \ and \ \ f : \Omega \rightarrow R \end{aligned} \tag{2.1}$$

Global optimization: in terms of maximization, given $f : \Omega \rightarrow R$, $\Omega \neq \emptyset$, for $\mathbf{x} \in \Omega$, the $f(\mathbf{x}^*)$ is the global maximum optimal solution if and only if

$$\forall f(\mathbf{x}) \in \Omega : \quad f(\mathbf{x}^*) \geq f(\mathbf{x}) \tag{2.2}$$

Optimization problems can be grouped according to their complexity. Some complex optimization problems have been proved to be nondeterministic-polynomial-time (*NP-hard*) [Johnson, 1985], which means that, there is no solution method (so far) able to solve them in polynomial time [Papadimitriou e Steiglitz, 1982]. For example, the *Traveling Salesmen Problem*

[1] In the literature, problems variables are also called decision variables, or only variables.

(TSP) [Dorigo e Gambardella, 1997, Hoffman et al., 2013] is a a *NP-hard* problem, where there exists a finite number of cities, and given the distances between each pair of cities. The objective is to find the shortest possible route that visits each city only once and returns to the origin. Its solutions $\mathbf{x}$ is represented as a *permutation* in the *search space*. For small scale problems such as 4 cities, it is possible to compute all the different permutations to determine the best one, whereas for 100 cities, it is considered impossible to be solved on a reasonable computational time.

Exact methods (deterministic methods) are not preferable to address these problems because exact algorithms might need exponential computational time to find the optimal solution in the worst case. This often leads to high computational cost for practical purposes [Papadimitriou e Steiglitz, 1982]. Even though we do not know how the optimal solution looks like, usually, in optimization problems, we can assess the quality of a candidate solution, and use this information to guide the search. Therefore, *approximation methods* have received much more attention. *Approximation methods* do not guarantee the optimal solution, but they can find an approximated optimal solution in a reasonable computational time. For some complex problems, approximated optimal solutions have been considered acceptable by the users of the application.

Various approximation methods have, as their primary concept, the search in a given neighborhood. This concept is called *local search* [Glover, 1986]. Let a neighborhood of $\mathbf{x}$ be $\mathcal{N}(\mathbf{x})$. A solution $\mathbf{x}'$ is a neighbor solution, i.e., $\mathbf{x}' \in \mathcal{N}(\mathbf{x})$ if $\mathbf{x}'$ can be achieved by a single *move* from $\mathbf{x}$. A basic *Local search* algorithm starts from some initial solution and iteratively tries to replace the current solution by a better solution until a local optimum is reached. It means that, these kind of algorithms enables guide the search to a local optimal region of the *search space*. A *locally minimum* with respect to a neighbor structure $\mathcal{N}$ is a solution $\mathbf{x}^+$, such that $\forall \mathbf{x}' \in \mathcal{N}(\mathbf{x}) : f(\mathbf{x}^+) \leq f(\mathbf{x}')$. This concept of search in the neighborhood is also called as *hill-climbing* [Blum e Roli, 2003, Luke, 2009] because they iteratively test new candidate solutions in the region of the current candidate, and adopts the new candidate if it is better regarding the quality assessment.

In addition, the *approximation methods* can be distinguished between *constructive* and *perturbative* methods [Blum e Roli, 2003]. Constructive algorithms initialize the solutions empty, and then add solution components until they reach a complete solution. Perturbative algorithms start from a (usually, random) initial solution and iteratively test new solutions by perturbing the current solution taking into account the structure of the search space.

Usually, for solving hard optimization problems, these simple methods do not guarantee at finding the global optimal solution(s) because the problem may contain several local optimal regions. Figure 2.1 describes three different trajectory landscape, where each sub-figure has different number of local optimum and objective function landscape throughout the search. A simple trajectory method may guide the search to a local optimum [Blum e Roli, 2003, Luke, 2009].

Since the 80's, more sophisticated *approximation methods* have been proposed, and they are commonly called metaheuristics [Glover, 1986, Blum e Roli, 2003, Glover e Kochenberger, 2006]. In the literature, researchers have defined metaheuristics in different manners. For instance:

- "*A metaheuristic is formally defined as an iterative generation process, which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space. Learning strategies are used to structure information in order to find efficiently approximated optimal solutions*" [Osman e Laporte, 1996].

- "*Metaheuristics are typically high-level strategies which guide an underlying, more problem specific heuristic, to increase their performance. The main goal is to avoid the*

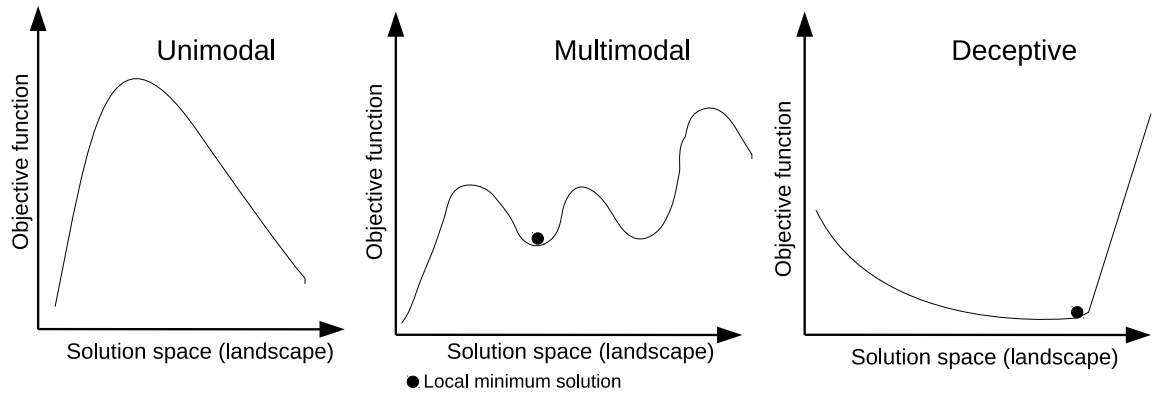


Figure 2.1: Three examples of function landscape regarding the trajetory in the search space

disadvantages of iterative improvements and, allowing the local search to escape from local optima. This is achieved by allowing worsening moves or generating new starting solutions for the local search in a more intelligent fashion than just providing random initial solutions" [Stützle, 1998].

- "*Metaheuristics are high level strategies for exploring the search spaces by using different methods, balancing the exploration and exploitation. This is important, on one side to quickly identify regions in the search space with high quality solutions, and on the other side not to waste too much time in regions which are either already explored or which do not provide improved solutions*" [Blum e Roli, 2003].

- "*Metaheuristics are a class of stochastic optimization algorithms and techniques, which employ some degree of randomness trying to explore the search space aiming to find the optimal solution to the optimization problem in question*" [Luke, 2009].

Therefore, one essential concept for developing efficient metaheuristics is to incorporate mechanisms to avoid getting trapped in local optimal regions of the *search space*. Some metaheuristics separate the *problem* and the *solver* (thus, they are, usually, called as *black-box methods*). However, metaheuristics may take into account some problem-specific knowledge in the form of heuristics that are controlled by the high level strategy. For example, in TSP, the distance between two cities can be used as a heuristic (local) information, and thus be a useful component to guide the search [Blum e Roli, 2003, Rothlauf, 2011].

In short, we say that the primary goal of a metaheuristic algorithms is to explore the *search space* efficiently, guiding the algorithm for finding the global optimal solution. This goal is directly related by the dynamic balance between *diversification* and *intensification*. In this thesis, we call this relation as *algorithm search ability*, and *diversification* is the search ability to *explore* different regions of the *search space* while *intensification* is the ability to *exploit* (search in the neighborhood) of the promising solution, and thus to keep converging towards the optimum.

Figure 2.2 shows an overview on metaheuristic classification. In this classification, the metaheuristics can be distinguished regarding the number of solutions used at the same time. Thus, we have the *single-solution* methods and *population-based* methods. Examples of single-solution methods are: Simulated Annealing (SA) [Brooks e Morgan, 1995], Tabu Search [Glover, 1989], Iterated local search (ILS) [Glover e Kochenberger, 2006], Greedy randomized adaptive search procedure (GRASP) [Feo e Resende, 1995], and Variable neighborhood search (VNS) [Hansen e Mladenović, 2001, Hansen e Mladenović, 2003]. These metaheuristics dynamically explore the neighborhood of the current solution, but also incorporate a strategy to

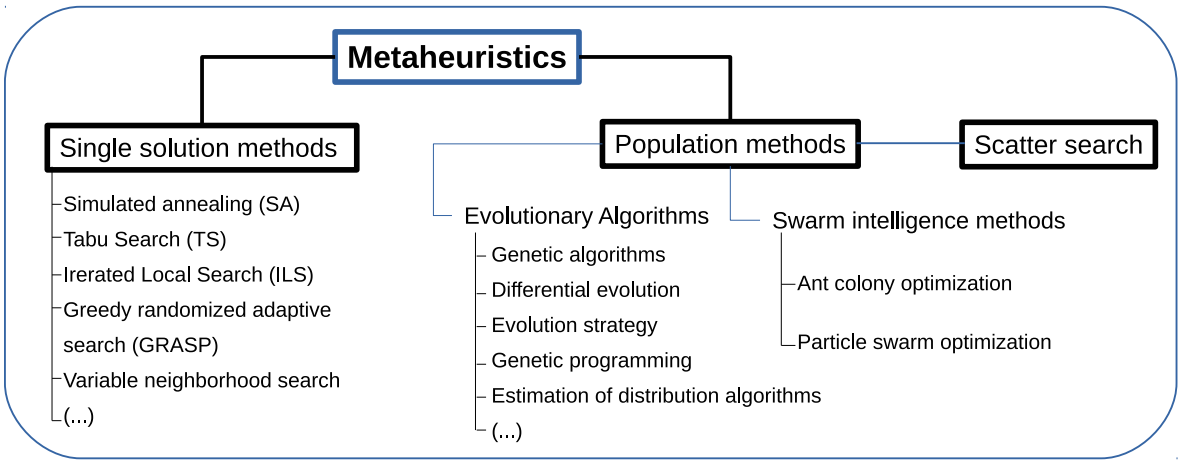


Figure 2.2: Overview of some metaheuristic techiniques

avoid the local optima. It means that, sometimes, a new candidate solution may not belong to the neighborhood of the current candidate solution intending to obtain a more effective exploration [Blum e Roli, 2003]. Example of these strategies ares: (i) the restart mechanisms (i.g, to restart a solution randomly if a local optimal is reached), (ii) to adopt solutions with worst quality than the current one with a certain probability (*acceptance criterion*), (iii) the use of a *short memory* to avoid moving to solutions already evaluated, (iv) shaking mechanism (i.e., a heavy perturbation in the current solution to provide a good restart solution placed in other region of the *search space*, and (v) to iteratively change the neighborhood structure.

Different from the single-solution methods, some metaheuristics work with a set of candidate solutions at the same time, these metaheuristics are classified under the umbrella of *population-based* methods. Most of these methods are inspired by some behavior from nature. Examples of these metaheuristics are: the Evolutionary Algorithms (EAs), the ACO and the EDAs. As they deal with a population of solutions, *population-based* algorithms provide a natural approach to explore different regions of the search space.

As well as, population-based methods and single-solution methods can be combined into a unique metaheuristic. Usually, these metaheuristics are called hybrid methods [Knowles e Corne, 2005, Blum e Roli, 2008]. The idea is to use the concepts of both strategies to efficiently balance the *exploration vs. exploitation*.

2.2 Evolutionary Algorithms

EA is a set of metaheuristics based on the Darwinian concept of *survival of the fitness*. EAs computationally simulate a natural evolution process where the solutions (*individuals*) with higher quality function (*fitness*) has more probability to survive and then to reproduce new solutions for the next generations [Coello et al., 2007].

Holland [Holland, 1975] proposed the *genetic algorithm* (GA), and it was initially applied to solve combinatorial problems with binaries domains. In a basic GA, two or more *parent individuals* are selected from the current population according to a elitism preserve criterion, and then the *reproduction step*[2]) (based on recombination and mutation) generates the new individuals (*offspring*). The new individuals are used to update the current population. Figure 2.3 represents the basic genetic operators based on *one-point crossover* and *one-bit-flip*

[2]The reproduction step is also called *variation step*.

mutation for combinatorial problems with binary representation. In this figure, two solutions are selected and recombined to generate an offspring.

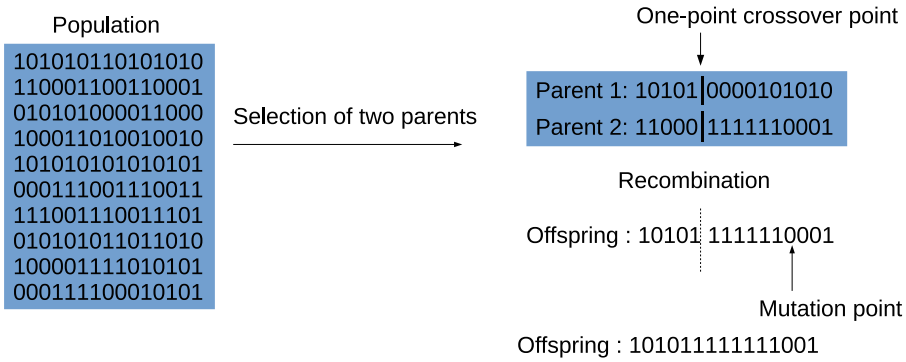


Figure 2.3: Representation of a population of solutions with binary domain, one-point crossover and one-bit-flip mutation

Algorithm 2.1 presents the pseudo-code of a generational EA. First, the population of encoded solutions is initialized randomly, and evaluated by the objective function that measures their quality according to the target problem. Next, the algorithm, according to a *elitism persevere mechanism*, selects μ individuals for the *reproduction* step. The selected population (S) is manipulated by a set of operators to generate new candidate solutions (P'). The new individuals are used to update the current population. The solutions with highest fitness have more probability to survive in the next *generation*. The algorithm stops when a condition is met and returns the best solution found so far.

Algorithm 2.1: General evolutionary algorithm framework

```
1 Generate initial population P
2 For each x ∈ P compute its fitness assessment.
3 while a termination condition is not met do
4 |   S ← Select μ solutions from P according to a elitism preserve mechanism.
5 |   P' ← Reproduction(S) % Apply genetic operators to generate new offspring
6 |   For each x' ∈ P' compute its fitness assessment.
7 |   P ← Update the current population using P'
8 end while
9 Return the solution with the best fitness assessment.
```

Usually, EAs with the inclusion of a LS procedure are called *memetic* algorithms [Knowles e Corne, 2005, Blum e Roli, 2008]. In general, a LS procedure can be applied after the reproduction step, trying to explore the neighborhood of the new offspring.

2.3 Ant Colony Optimization

According to [Battiti et al., 2008], the *reactive search optimization*, i.e., the *learning while optimizing* is a principle that have been widely accepted as a basic design in metaheuristics. This principle integrates machine learning techniques into heuristics and self-tunes algorithms behavior based on on-line information collected during the search. Ant Colony Optimization (ACO) proposed by [Dorigo et al., 1996] adopts this principle. It was originally proposed for solving combinatorial problems [Dorigo e Gambardella, 1997] [Stützle e Hoos, 2000].

ACO is a population-based metaheuristic inspired by the indirect communication of real ants through trails of a chemical substance called pheromone [Dorigo et al., 1996]. This behavior enables real ants to find shortest paths between food sources and their nest.

Computationally, the *artificial ants* are stochastic agents that use the information that reflects the experience accumulated by the previous artificial ants. ACO represents the knowledge learned throughout the generations as pheromone matrices, where the solutions components of the target problem can be represented as a graph. At current time t, the pheromone value of a solution component represents the probability "attractive" of this component be chosen (added) by an artificial ant during the construction process. Moreover, ACO algorithms allow the incorporation of problem-specific knowledge (heuristic) during the construction step. According to [Dorigo et al., 2006, Dorigo et al., 2008] these principles provide a common framework for the most applications of ACO to solve combinatorial optimization problems.

Different from the genetic operators (which are based on recombination and perturbation), in ACO algorithms, each variable x_i is defined one by one (i.e., constructive method). First, the *candidate solution* is initialized empty, and then the algorithm iteratively (and stochastically) adds a solution component until a complete and feasible solution is generated.

Algorithm 2.2 presents the general ACO framework. First, the parameters and pheromone matrices τ are initialized according to a start value τ_0. At each generation, a number of ants construct new solutions by adding solution components x_i based on (i) the current *pheromone matrix* (τ) and (ii) *the heuristic information* (η) which are combined to define a *probabilistic rule*. In a probabilistic rule, the influence of τ and η can be balanced by their respective coefficients. Different forms of defining the probabilistic rules have been considered [Dorigo et al., 2008, Mohan e Baskaran, 2012, López-Ibánez e Stutzle, 2012]. After an artificial ant ends its construction solution and calculates their quality functions, the best solution(s) found so far are determined. Finally, each pheromone amount of τ is updated according to the quality assessment of the best solution(s) found so far. The algorithm stops when a termination condition is met.

Algorithm 2.2: General ACO

```
1 Parameters Initialization
2 while termination condition not met do
3 |   ConstructAntSolutions
4 |   ApplyLocalSearch %optional
5 |   UpdatePheromone
6 end while
```

Another characteristic in ACO algorithms is that, in order to prevent unlimited accumulation of the pheromone amount, typically during the pheromone update, all pheromones trails are decreased by a factor that models the *evaporation*. In the *Max-Min* Ant System [Stützle e Hoos, 2000], the pheromone trails are bounded to an interval $[\tau_{min}, \tau_{max}]$, and all the pheromone trails are initialized according to the maximum value τ_{max}. These explicit restrictions prevent the premature convergence, and thus leading to exploration. Optionally, at each generation, local search procedures can be used to improve the solutions.

ACO algorithms have been applied to solve a broad range of combinatorial optimization problems [Dorigo et al., 2006, Dorigo et al., 2008, Mohan e Baskaran, 2012, López-Ibánez e Stutzle, 2012]. Besides its successful application in the literature, ACO algorithms can suffer from several parameters to be set a priori, and their high computational cost in comparison to other metaheuristics, such as GA [Dorigo et al., 2008].

2.4 Estimation of Distribution Algorithms

EDAs [Mühlenbein e Paass, 1996, Larrañaga e Lozano, 2002, Pelikan et al., 2007] are a set of EAs characterized by the use of explicit probability distributions to explore the *search space*. By incorporating machine learning techniques into evolutionary algorithms, EDAs were originally proposed to solve some complex problems, in which traditional genetic operators are not very successful. The motivation is that, usually, genetic operators fail in automatically identifying the structure of the complex problem, which consists of several correlated variables. This problem, is referred in the literature as the *linkage-learning* [Mühlenbein e Paass, 1996].

Algorithm 2.3 presents the general steps of an EDA. In this pseudo-code, P is the population, S is the set of selected promising solutions. First, the algorithm generates the initial population and computes their respect fitness assessment. Next, it learns a probabilistic model (p) from the set of selected solutions (S) trying to explicitly express the interrelations between the problem variables. The offspring are then obtained by sampling the probabilistic model learned. The sampled solutions (P') are used to update the current population. The algorithm stops when a particular condition is met such as a maximum number of generations or a limited computational cost.

Algorithm 2.3: Estimation of Distribution Algorithm (EDA)

```
1 Generate initial population P
2 For each x ∈ P compute its fitness assessment.
3 while termination condition not met do
4 |   S ← Select μ solutions from P according to a elitism preserve mechanism.
5 |   p ← Build a probabilistic model from S.
6 |   P′ ← sample μ new solution(s) from p.
7 |   For each x′ ∈ P′ compute its fitness assessment.
8 |   P ← Update the current population using P′
9 end while
```

[Pelikan et al., 2007] provide two motivations for using EDAs instead of EAs based on genetic operators:

- **Motivation from genetic operators:** EDAs enable simultaneous exploration of multiple regions in the *search space* and allows the use of statistical and learning techniques to identify problems regularities automatically, and express them in an explicit manner. The regularities provided by them can reach the global optimal solution, where genetic operators and local search methods can easily fail due to exponentially many local optima and strong large-order interactions between the variables.

- **Motivation from Machine Learning:** The use of probabilistic models to guide the search enables the application of different statistical modeling and sampling techniques to automatically discover, and exploit problem regularities for effective exploration. In most EDAs, their probabilistic models are represented by a *graphical model*, which combines graph theory, modularity and statistics to provide an approach for *learning* and *sampling* probability distributions.

Likewise, in EDAs, the form in which the *learning* and *sampling* components are implemented are critical for their performance and computational cost. According to [Larrañaga et al., 2012], one of the rationales in EDA development has been to find a satisfactory

trade-off between the complexity of the probabilistic models and how accurately these models represent particular optimization problems characteristics.

The probabilistic models can be categorized according to their strategy to provide a probability distribution as (i) *univariate models* and (ii) *multi-variate models*. In the following, we introduce them.

Univariate probabilistic models:

First of all, we define some terminologies. Let p represent a vector of positive distributions, $p(x_I)$ denote the marginal probability for $\mathbf{X}_I = \mathbf{x}_I$, and let $p(x_j \mid x_k)$ denote the conditional probability distribution of $X_j = x_j$ given $X_k = x_k$.

In the univariate marginal distribution (or univariate probabilistic model), the variables are considered to be independent, and the probability of a solution is the product of the univariate probabilities for all the variables:

$$p_u(\mathbf{x}) = \prod_{j=1}^{n} p(x_j) \tag{2.3}$$

One of the simplest EDAs that use the univariate model is the univariate marginal distribution algorithm (UMDA) [Mühlenbein e Paass, 1996]. UMDA uses a probability vector $p_u(\mathbf{x})$ as the probabilistic model, where $p(x_j)$ denotes the univariate probability associated to the corresponding discrete value. Let assume that this EDA is for combinatorial problems with binary representation $\{0, 1\}$. To learn the probability vector for these problems, each $p(x_j)$ is set to the proportion of "1s" in the selected population S. To generate new solutions, each variable is independently sampled.

The population-based incremental learning (PBIL) [Baluja, 1994], like UMDA, uses the probabilistic model in the form of a probability vector $p_u(\mathbf{x})$. The initial probability of a "1" in each position $p(x_j)$ is set to 0.5. The probability vector is updated using each selected solution $\mathbf{x}$ in S. For each variable, the corresponding entry in the probability vector is updated by:

$$p(x_j) = (1.0 - \alpha)p(x_j) + (\alpha * x_j) \tag{2.4}$$

where α is the learning rate specified by the user.

At each generation, during the *sampling*, each position of the vector solution has a small probability to be perturbed based on a mutation rate parameter to prevent premature convergence [Baluja, 1994, Pelikan, 2011].

Multi-variate probabilistic models:

Often the variables in a problem are related in some manner. Therefore, EDAs that assume dependencies between the variables have been proposed. In this case, the probabilistic model is capable of capturing some *pair-wise* interaction between the variables. The probability distribution is represented by a probabilistic graphical model (PGM) which maintains joints (conditional) probabilities.

Tree-based models [Pelikan e Mühlenbein, 1999] are a kind of PGMs capable to capture some *pair-wise* interactions between variables. In a tree model, the conditional probability of a variable may only depend on at most one other variable, being its parent in the tree structure.

The probability distribution $p_{\mathcal{T}}(\mathbf{x})$ that is conformal with the tree model is defined as:

$$p_{\mathcal{T}}(\mathbf{x}) = \prod_{j=1}^{n} p(x_j|pa(x_j)), \tag{2.5}$$

where $pa(X_j)$ is the parent of X_j in the tree, and $p(x_j|pa(x_j)) = p(x_j)$ when $pa(X_j) = \emptyset$, i.e. X_j is a root of the tree.

The bivariate marginal distribution algorithm (BMDA) proposed in [Pelikan e Mühlenbein, 1999] uses a model based on a set of mutually independent trees (a forest)[3]. In each generation of the algorithm, a tree model is created and sampled to generate new candidate solutions based on the conditional probabilities learned from the population.

The learning process from the Tree-EDA proposed in [Santana et al., 2001b] combines features from algorithms introduced in [Baluja e Davies, 1997] and [Pelikan e Mühlenbein, 1999]. In Tree-EDA, the learning procedure works as follows:

- Step 1: Compute the univariate and bivariate marginal frequencies $p_j(x_j)$ and $p_{jk}(x_j|x_k)$ using the set of selected promising solutions S;
- Step 2: Calculate the matrix of mutual information using the univariate and bivariate frequencies;
- Step 3: Calculate the maximum weight spanning tree from the mutual information. Compute the parameters of the model.

The idea is that by computing the maximum weight spanning tree from the matrix of mutual information, the algorithm will be able to capture the most relevant bivariate dependencies between the problem variables. Details on EDAs that use the tree models can be obtained from [Baluja e Davies, 1997, Pelikan e Mühlenbein, 1999]. More details on the use of PGMs for probabilistic modeling in EDAs can be obtained from [Larrañaga et al., 2012].

Some researchers have been dealing with finding similarities between ACO algorithms and EDAs such as PBIL and UMDA [Blum e Roli, 2003]. Furthermore, traditional approaches to learn a model from a set of selected solutions in EDAs give the same importance to all solutions in the set, and thus learn marginal frequencies without distinguishing the fitness quality. However, it makes sense using the fitness information of the selected solutions at the time of learning. *Fitness-aware* model learning refers to methods that incorporate information about the fitness evaluation values during learning of the probabilistic model [Mühlenbein et al., 2003]. Usually, this scheme is called Boltzmann distribution [Mühlenbein et al., 2003, Valdez et al., 2013].

2.5 Multi-objective optimization

Until now, we have presented an overview of optimization and metaheuristics considering the single-objective case. Besides, some real-world problems may contain more than one objective to be optimized, and usually these objectives are in conflict with each other. This section is devoted to define MOPs, and how they are solved using metaheuristics.

MOPs can be defined as the problem of finding, not a unique solution, but a set of solutions which satisfy constraints and optimize a vector whose elements represent the objective functions [Miettinen, 2012].

$F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x}))^T$ represents a vector of objective functions, where m is the number of objective functions to be optimized. In MOPs, the optimization of m objective functions may involve the maximization of all m functions, the minimization of all m functions, or a combination of both [Miettinen, 2012].

[3]For the convenience of notation, this set of mutually independent trees is referred as Tree.

Therefore, different from the single-objective optimization problems, in this case, the *objective space* is multi-dimensional, defined as R^m, where each coordinates axis correspond to a objective function. Thus, each feasible solution in the *search space* ($\mathbf{x} \in \Omega$) maps a $F(\mathbf{x})$ in the *objective space*, where $F : \Omega \rightarrow R^m$ and $F(\mathbf{x})|\mathbf{x} \in \Omega$.

The mathematical definition of a MOP can be stated as follows:

$$\begin{aligned} &\textit{minimize (or maximize) } F(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_m(\mathbf{x})) \\ &\textit{subject to } \mathbf{x} \in \Omega \end{aligned} \tag{2.6}$$

It is a misconception, in practice, to design this kind of problems into a single objective problem to provide an equivalent cost or a profit value [Miettinen, 2012]. The different objectives are typically non-commensurable, and it is hard to aggregate them into one synthetic objective. Converting a multi-objective optimization problem into a simple single-objective problem puts the decision maker (DM) before optimization. Besides, if no DM preference is known a priori, all the objectives are considered equally important. So, the set of "equally" good solutions has to be defined in some form. The science community represents this set of optimal solutions through the use of Pareto Optimality Theory [Pareto, 1964, Ehrgott, 2006].

Pareto optimality[4] [Pareto, 1964, Ehrgott, 2006]: Let $\mathbf{x}, \mathbf{y} \in \Omega$, $\mathbf{x}$ is said to *dominate* $\mathbf{y}$ if and only if $f_l(\mathbf{x}) \leq f_l(\mathbf{y})$ for all $l \in \{1, .., m\}$ and $f_l(\mathbf{x}) < f_l(\mathbf{y})$ for at least one l. A solution $\mathbf{x}^* \in \Omega$ is called *Pareto optimal* if there is no other $\mathbf{x} \in \Omega$ which *dominates* $\mathbf{x}^*$. So, any improvement in one objective in $F(\mathbf{x})$ must lead to a deterioration of at least another objective. The set of all the *Pareto* optimal solutions is called the *Pareto set (PS)* and the solutions mapped in the *objective space* are called *Pareto front (PF)*, i.e., $PF = \{F(\mathbf{x})|\mathbf{x} \in PS\}$.

In many real-life applications, the *PF* is of great interest to decision makers (DM) for understanding the trade-off nature of the different objectives and selecting their preferred final solution [Miettinen, 2012]. The true *PF* of a problem may contain a large set or infinite number of Pareto solutions. Thus, achieving the exact true *PF* from a complex problem is, usually, quite difficult. Nevertheless, reasonable good approximations of the *true PF* (within a limited computational time) are generally acceptable by DMs. Multi-objective optimization algorithms attempt to find these approximated *PFs*. In other words, an accurate multi-objective optimization algorithm should be able, throughout the search, to converge towards the *true PF*, while exploring different regions of the search space.

In the literature, single-solutions methods have been applied to solve MOPs, such as the multi-objective tabu search [Hansen, 1997, Gandibleux e Freville, 2000], and the multi-objective simulated annealing [Smith et al., 2004]. These methods instead of guide the search using a unique (best) solution, the algorithms use a set of non-dominated solutions found so far. On the other hand, population-based methods (here we also include hybrid approaches) are suitable for MOPs because they deal simultaneously with a set of possible solutions (*population*), which allows finding several non-dominated solutions in a single *run* [Coello et al., 2007, Miettinen, 2012].

According to [Coello et al., 2007], the primary goals of a multi-objective evolutionary algorithm (MOEA) are:

1. Preservation of non-dominated solutions according to the Pareto dominance concept;
2. Progression towards the PF_{true} in the *objective space* throughout the generations;
3. Diversity of the approximated *PF*, i.e., capability of the algorithm to explore the different regions of the *objective space* for finding solutions that cover the entire PF_{true};

[4] This definition of domination is for minimization. For maximization, the inequalities should be reversed.

4. Representation of a reasonable number of approximated *Pareto* optimal solutions for the DM.

Several MOEAs have been proposed in the literature. Nowadays, they can be broadly categorized under the primary method to maintain the set of non-dominated solutions [Deb e Jain, 2014]. They are defined as follows:

- **Dominance-based method:** In this method, a MOP is optimized by simultaneously optimizing all the objectives. The assignment of fitness to solutions is based on Pareto-dominance principle which plays a key role in the convergence towards *PF*. According to [Li e Zhang, 2009], using the Pareto dominance alone could discourage the diversity of search. Thus, an explicit diversity preservation scheme is necessary to maintain diversity. Some techniques such as *fitness sharing* and *crowding distance* have often been incorporated as compensation in the Pareto-based algorithms. Some of the remarkable Pareto-based MOEAs are the NSGAII [Deb et al., 2002], the SPEA2 [Zitzler et al., 2001].

- **Indicator-based method:** In this method, an indicator assessment such as the Hypervolume (see Section 2.7.1) is used to measure the fitness of a solutions by assessing its contribution to the combined convergence and diversity measure. The IBEA and the SMS-EMOA [Beume et al., 2007] are examples of Indicator-based MOEAs.

- **Decomposition-based method:** In this method, a MOEA uses a scalarizing aggregation function, such as the *Weighted Sum*, and a set of weight vectors to convert the MOP in question into a number of single-objective optimization subproblems. The decomposition-based MOEAs utilize the aggregated fitness value of the solutions to select the best solutions. Representative MOEAs of this kind are the MOGLS [Ishubuchi e Murata, 1998], and a wide range of algorithms based on MOEA/D framework [Zhang e Li, 2007]. In MOEA/D, the subproblems are optimized simultaneously in a collaborative manner using the concept of neighborhood between the subproblems.

Since Zhang et al. [Zhang e Li, 2007] proposed the MOEA/D framework, the decomposition-based method have become more and more popular [Li et al., 2015, Trivedi et al., 2016]. Besides, some algorithms can combine two methods, for instance, the NSGA-III [Deb e Jain, 2014], which is considered both Pareto-based and decomposition-based. This algorithm was proposed to address the issues that emerge from problems that have great number of objectives (e.g. more than three objectives) and different Pareto shapes [Li et al., 2015].

In this thesis, we have taken a research direction, we have decided to give focus only to decomposition-based MOEA because MOEA/D variants have become a recent research trend in optimization [Trivedi et al., 2016]. Therefore, in the following, we describe the general MOEA/D framework that serves as a basis for this work.

2.6 MOEAs based on Decomposition

A Pareto optimal solution to a MOP could be an optimal solution of a single objective optimization problem in which the objective is an aggregation function of all the objectives. Thus, the *PF* of a MOP can be decomposed into a number of single objective optimization subproblems. This concept is the core idea behind many traditional mathematical programming methods for approximating the *PF* [Li e Zhang, 2009, Miettinen, 2012].

The MOEAs based on decomposition uses a specific scalarizing aggregation function with a set of weight vectors to decompose a MOP into a number of single-objective optimization

subproblems and optimize them in a collaborative manner. Since the last decade, algorithms such as MOGLS [Ishubuchi e Murata, 1998] and a wide range of algorithms based on the MOEA/D framework [Zhang e Li, 2007] have been proposed for solving different continuous and combinatorial MOPs. This method is suitable to solve multi-modal problems, in which they explore different regions of the search space.

Let $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_m)$ be a weight vector, where $\sum_{l=1}^{m} \lambda_l = 1$, $\lambda_l \geq 0$ for all $l = 1, ..., m$, and m is the number of objectives.

The well known MOEA/D framework [Zhang e Li, 2007] decomposes a MOP into (N) scalar single-objective optimization subproblems. Each subproblem i is associated to a weight vector $\boldsymbol{\lambda}^i$, and the set of all weight vectors is $\{\boldsymbol{\lambda}^1, ..., \boldsymbol{\lambda}^N\}$.

The distance between any two weight vectors is defined according to the Euclidean distance between them. The neighborhood for each subproblem i, is defined as $B(i) = \{i_1, ..., i_T\}$ where T is the neighborhood size. Any information about T weight vectors (solutions) close to λ^i should be helpful for optimizing $g(\mathbf{x}|\boldsymbol{\lambda}^i)$. This is the major motivation behind MOEA/D [Zhang e Li, 2007].

$B(i)$ is used for the *selection* of parent solutions and the *update* of the current solutions. The size of the neighborhood T used for *selection* and *update* plays a vital role in MOEA/D to exchange information among the subproblems. Moreover, the trade-off between diversity and convergence can be controlled by the different mechanisms and parameters of the algorithm [Li e Zhang, 2009, Wang et al., 2014, Li et al., 2015].

Algorithm 2.4 presents the pseudo-code of a general MOEA/D which serves as a basis for this thesis.

Algorithm 2.4: General MOEA/D framework

```
1  Initialize the N weight vectors λ = (λ^1, ..., λ^N)
2  Initialize population Pop = (x^1, ..., x^N) randomly or by a problem-specific method and
     compute every F(x^i)
3  Initialize EP with the non-dominated solutions from Pop
4  while a termination condition is not met do
5      for each subproblem i ∈ 1, ..., N at each generation do
6          y ← Variation(B(i));
7          Compute F(y);
8          Update_Neighborhood(B(i), y);
9      end for
10     UpdateEP(Pop, EP)
11 end while
12 Return Pop, EP
```

Initialization step: The N uniform distributed weight vectors $\boldsymbol{\lambda}^1, .., \boldsymbol{\lambda}^N$ are set. Usually, as in [Zhang e Li, 2007] the weight vectors are generated using a control parameter H, where an element of each weight vector (λ_l^i) is one of the $\{0/H, 1/H, ..., H/H\}$ and N is defined according to the combinatorial $C_{H+m-10}^{m-1} = N$. Then, the Euclidean distance between any two weight vectors λ^i, λ^j is computed. For each subproblem i, the set of neighbors $B(i)$ is initialized with the T closest neighbors. The initial population $Pop = (\mathbf{x}^1, ..., \mathbf{x}^N)$ is generated randomly or by a problem specific method. Then, their corresponding fitness functions $F(\mathbf{x}^1), ..., F(\mathbf{x}^N)$ are computed. The external *Pareto* (EP) is initialized with the non-dominated solutions from the initial population Pop.

Variation step: For each subproblem i, the reproduction is performed using $B(i)$ as the selected population. The conventional MOEA/D selects two parent solutions $p_1, p_2 \in B(i)$ and applies the genetic operators (*crossover* and *mutation*) to generate $\mathbf{y}$.

Update Neighborhood: The new solution $\mathbf{y}$ is used to update the parent population. $\mathbf{y}$ replaces the current solutions $\mathbf{x}^r$ (where r is an index subproblem from in $B(i)$) if $\mathbf{y}$ has a better scalar aggregation function value ($g(\mathbf{y}|\lambda^r)$) than $g(x^r|\lambda^r)$. In addiction, different update schemes can be used, such as (i) the global scheme, where $\mathbf{y}$ can update any solution in Pop [Wang et al., 2014] instead of the solutions bounded by $B(i)$, and (ii) maximum number of replacements by $\mathbf{y}$, which prevents one solution having many copies in the current population.

Update EP step: The external population EP is used to maintain all non-dominated solutions found so far during the search. This step removes from EP all solutions dominated by $\mathbf{y}$ and adds $\mathbf{y}$ if no solution dominates it.

Likewise, EP can be optional. Thus the outcome of the algorithm can be the N final solutions. If the set of weight vectors is well distributed along the PF_{true} of the problem, the algorithm may be able to find distinct distributed solutions that cover the PF_{true}.

Several scalarizing aggregation functions have been used in the decomposition approach. The *Weighted Sum*, the *Tchebycheff*, and the *Penalty-based Boundary Intersection* (*PBI*) are the most common aggregation functions used [Miettinen, 2012]. In the following, these decomposition approaches are described:

Weighted Sum Approach: The optimal solution to the following scalar single-optimization problem is defined as:

$$\begin{aligned} \textit{minimize (or maximize) } g^{ws}(\mathbf{x}|\boldsymbol{\lambda}) = \sum_{l=1}^{m} \lambda_l f_l(\mathbf{x}) \\ \textit{subject to } \mathbf{x} \in \Omega \end{aligned} \tag{2.7}$$

which $g^{ws}(\mathbf{x}|\boldsymbol{\lambda})$ is to be minimized if the MOP is minimization, and $g^{ws}(\mathbf{x}|\lambda)$ is to be maximized if the MOP is maximization.

To generate a set of different Pareto optimal solutions, we can use different weight vectors λ in the *Weighted Sum*. If we use a well distributed set of weight vectors, and the shape of PF, from the MOP in question, is concave (convex), then the *Weight Sum* could work well.

Tchebycheff approach: The optimal solution to the following scalar single-optimization problem is defined as:

$$\begin{aligned} \textit{minimize } \; g^{tc}(\mathbf{x}|\boldsymbol{\lambda}, \mathbf{z}^*) = \max_{1 \le l \le m} \{\lambda_l |f_l(\mathbf{x}) - z_l^*|\}; \\ \textit{subject to } \mathbf{x} \in \Omega \end{aligned} \tag{2.8}$$

where $\mathbf{z}^* = (z_1^*, ..., z_m^*)^T$ is the reference point (also called ideal vector), i.e., $z_l^* = \max\{f_l(\mathbf{x})|\mathbf{x} \in \Omega\}$ for each $l = 1, ..., m$. For each Pareto optimal solution $\mathbf{x}^*$ there exists a weight vector $\boldsymbol{\lambda}$ such that $\mathbf{x}^*$ is the optimal solution of Eq. (2.8), and each optimal solution of Eq. (2.8) is a Pareto optimal solution of Eq. (2.6).

In *Tchebycheff approach*, the optimization of $g^{tc}(\mathbf{x}|\boldsymbol{\lambda}, \mathbf{z}^*)$ is always minimization, because it is related to the approximation of $F(\mathbf{x})$ to the reference point $\mathbf{z}^*$ that contains the best values found so far for each $f_l(\mathbf{x})$. Thus, $g^{tc}(\mathbf{z}^*|\boldsymbol{\lambda}, \mathbf{z}^*) = 0$.

The *PBI* approach is a variant of the normal-boundary intersection method, where equality constraint is handled by a penalty function. The approach is defined as follows:

$$minimize \;\; g^{pbi}(\mathbf{x}|\boldsymbol{\lambda}, \mathbf{z}^*) = d_1 + \theta d_2 \tag{2.9}$$

$$subject to \mathbf{x} \in \Omega \tag{2.10}$$

where

$$d_1 = \frac{||(F(\mathbf{x}) - \mathbf{z}^*)^T \boldsymbol{\lambda}||}{||\boldsymbol{\lambda}||} \tag{2.11}$$

$$d_2 = ||F(\mathbf{x}) - (\mathbf{z}^* + d_1 \frac{\boldsymbol{\lambda}||}{||\boldsymbol{\lambda}||})|| \tag{2.12}$$

where $\mathbf{z}^* = (z_1^*, ..., z_m^*)^T$ is the reference point, and $\theta \geqslant 0$ is a user-defined penalty parameter.

In PBI approach, d_1 is used to evaluate the convergence of $\mathbf{x}$ and d_2 measures diversity [Li et al., 2015]. The balance between d_1 and d_2 is controlled by θ. Li et al. [Li et al., 2015] present the different search behavior of PBI regarding the parameter θ. When θ is very close to zero (e.g., $\theta = 0.1$) its works as *Weighted Sum*. When $\theta = 1.0$, PBI works similar to *Tchebycheff*.

Current research on MOEA/D are various and include the extension of these algorithms to continuous MOPs with irregular Pareto shapes [Li e Zhang, 2009], many-objective optimization problems [Ishibuchi et al., 2013, Tan et al., 2013, Li et al., 2015, Ishibuchi et al., 2016], methods to parallelize the algorithm [Nebro e Durillo, 2010b], incorporation of preferences to the search [Pilat e Neruda, 2015], automatic adaptation of weight vectors (dynamic computational resource allocation) [Qi et al., 2014], automatically switching between different scalarizing aggregation functions [Ishibuchi et al., 2013], new strategies of selection and replacement to balance convergence and diversity [Wang et al., 2014], hybridization with local searches procedures [Alhindi e Zhang, 2014, Ke et al., 2014], hybridization with other metaheuristics such as ACO [Ke et al., 2013], and EDAs [Zangari et al., 2015b, Ma et al., 2016, Zapotecas-Martínez et al., 2015].

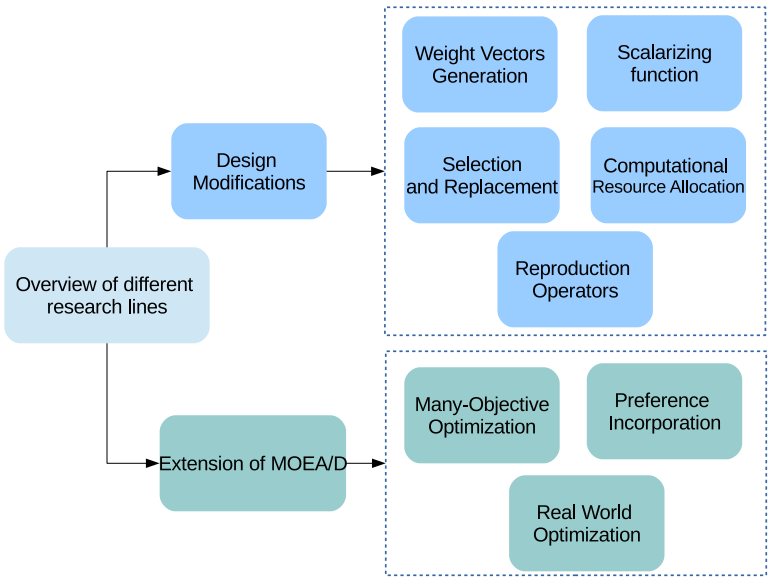


Figure 2.4: An overview of the current research lines on MOEA/D

Figure shows an overview of the current research lines on MOEA/D. Trivedi et al. [Trivedi et al., 2016] present a recent survey on MOEA/D. The authors differ the research lines along (i) design modifications in MOEA/D and (ii) extension of MOEA/D to solve specific problems. One of the challenges in MOEA/D algorithms is to determine the appropriated decomposition method for a particular problem [Trivedi et al., 2016]. Moreover, different parameters configurations have to be tested such as: (i) the scalarizing functions, (i) the form to distribute the weight vectors, (iii) the population size, (iv) the neighborhood size for selection and replacement. These MOEA/D components can deal with the balance between *exploration vs. exploitation*. As well, a tailored variation operator have to be used to achieve good results.

2.7 Experimental methodology for Multi-objective optimization

As mentioned before, the outcome of the multi-objective optimization algorithms can be an approximated Pareto-optimal set, which maps an approximated PF in the *objective space*. In this case, evaluate the quality of the outcome is more complex than in the single-objective case. The comparison of two approximated *PFs* gets even more complicated because some solutions in either set may be dominated by solutions in the other set, as also, some solutions of both sets are non-dominated [Zitzler et al., 2003].

The visual analysis of the *PFs* plotted in the *objective space* can be useful for two objectives (i.e., two-dimensional space), but it gets complex as the number of the objectives (coordinates) increases.

In the literature, different performance metrics[5] have been used to evaluate the multi-objective optimization algorithms [Zitzler et al., 2003, Zitzler et al., 2008, Coello et al., 2007]. These performance metrics attempt to indicate the quality of the *PFs* approximations. Usually, the performance metrics represent the quality of a approximated PF as a real number (unary indicators).

Knowles et al., 2006 state that several popular indicators are designed to asses just one isolated aspect of an PF approximations, such as the distance to the PF_{true}, the spread in *objective space*, or the evenness (sparsity) with which the solutions are distributed. The drawback of these different performance metrics is that a metric I_1 can indicate that the algorithm Alg_1 is better than Alg_2, whereas the metric I_2 indicates the contrary.

Knowles et al., 2006 claim that, one important property that an indicator should have is called *Pareto compliance*. It means that an indicator must not contradict the order induced by the Pareto dominance relation. The Pareto dominance relation of two approximated *PFs* is a relation extended from the definition of dominance of two vector solutions presented in Section 2.5, and it is described in the following.

Let A and B represent two approximated PFs. The symbol $\succeq$ represent the generalized approximation Pareto dominance relation between two Pareto sets A and B. Regarding that, $f_\tau(\mathbf{x}) \succeq f_\tau(\mathbf{y})$ means that $f_\tau(\mathbf{x})$ is not worse than $f_\tau(\mathbf{y})$ in all objectives. $A \succeq B \wedge B \not\succeq A$ means that every $\mathbf{y} \in B$ is dominated by at least one $\mathbf{x} \in A$. Then the indicator value of A must not be worse than the indicator value of B, i.e., $I(A) \geq I(B)$.

Several indicators that have been used in the literature are not *Pareto compliant*. Any indicator that can yield for any approximation sets $A, B \in \Omega$ a preference for A over B, when B is preferable to A with respect to Pareto dominance ($B \succeq B \wedge \neg A \succeq B$) is Pareto non-compliant [Knowles et al., 2006].

[5]In this thesis, *performance metrics* and *quality indicators* are synonymous, and they are used intercalated.

Additionally, if a stochastic multi-objective algorithm is applied several times to the same problem instance, a different approximation set may be returned each time. Due to this, the stochastic nature of the multi-objective algorithms have to be considered in an experimental study [Zitzler et al., 2008].

Therefore, in general, the **experimental methodology in multi-objective optimization** is based on the evaluation of the outcomes using two or more quality indicators and the application of statistical tests on the resulting distributions(s) obtaining. The comparison allows to state if there exist a significant difference between the algorithms or not. Thus, in a comparative study, any statement like "Alg_1 outperforms Alg_2" needs to be qualified by adding "according to the quality indicator I_1 and the statistical test."

Besides, the question of whether an algorithm outperforms another one involves various aspects such as the computational time required, the parameter setting, the benchmark test instances used, etc [Zitzler et al., 2008]. Usually, optimization algorithms have several parameters to be set, and thus preliminary experiments can be conducted to identify the best parameter values.

2.7.1 Performance metrics

The **Hypervolume Indicator** (I_{HV}) [Zitzler e Thiele, 1999] gives the volume of a space that is dominated by a *PF*. In order to measures the quantity, the *objective space* must be bounded, if it is not, then a bounding reference point that is dominated by all solutions must be used. Figure 2.5) represents the hypervolume.

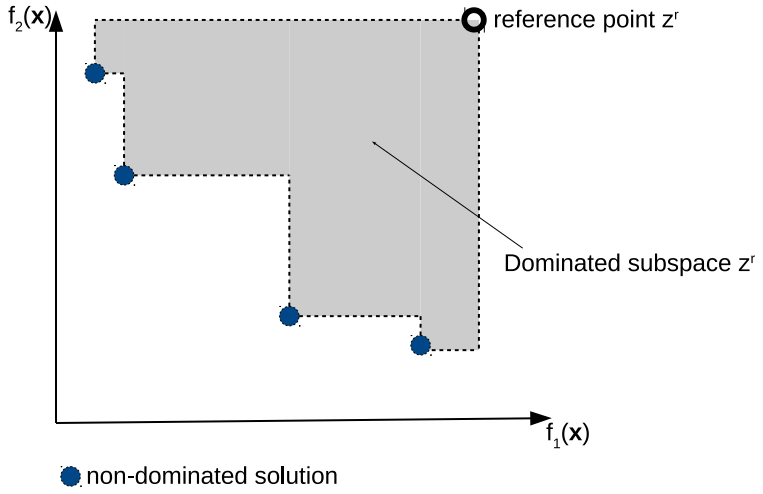


Figure 2.5: Illustration of the hypervolume indicator of a *PF*

Let P^* be a set of uniformly distributed *Pareto* optimal solutions along the *true PF* in the *objective space*, P be an approximated set to the *true PF* obtained by an algorithm considered, and $z^r = (z_1^r, \ldots, z_m^r)^T$ be a reference point in the objective space that is dominated by all Pareto-optimal objective vectors.

The I_{HV} measures the size (volume) of the objective space dominated by the solutions in P and bounded by z^r:

$$I_{HV}(P) = VOL(\bigcup_{\mathbf{x} \in P} [f_1(\mathbf{x}), z_1^r] \times \cdots \times [f_m(\mathbf{x}), z_m^r]) \tag{2.13}$$

The reference point (also called *nadir point*) is set to maximum values if the objective is minimization, and it is set to minimum values if the objective is maximization. Thus, the higher the $I_{HV}(P)$, the better the P in comparison to other fronts.

So far, the hypervolume indicator has been the only indicator proved to be *Pareto-compliant*. The drawback of I_{HV} is its computational cost that exponentially increases according to the number of objectives, and it is polynomial in the number of solutions in the approximation set. Furthermore, the reference point z^r needs to be specified a priori, and the indicator values are sensitive to the choice of this bound (scale invariant) [Zitzler et al., 2008]. The advantage of I_{HV} is that the PF_{true} is not needed.

I_{HV} is an unary indicator, but it can be extended to a binary indicator by defining $I_{HV}(A, B)$ as the hypervolume of the subspace of the *objective space* that is dominated by A but not by B [Knowles et al., 2006].

Cardinality (I_C): The cardinality of PF can be considered as an unary indicator. In this case, the indicator computes the number of solutions that compose the Pareto set. The I_C is cheap to compute, but it is not *Pareto compliant*.

Some indicators are only possible to compare pairs of sets, such as the Generational Distance (GD) and the Inverted Generational Distance (IGD). The IGD calculates how far the a approximated PF is from the PF_{true}.

IGD-metric (I_{IGD}) : The inverted generational distance from P^* to P is defined as

$$IGD(P^*, P) = \frac{\sum^{v \in P^*} d(v, P)}{|P^*|} \tag{2.14}$$

where $d(v, P)$ is the minimum Euclidean distance between v and the solutions in P. If $|P^*|$ is large enough $IGD(P^*, P)$ could measure both convergence and diversity of P in a sense. The lower the $IGD(P^*, P)$, the better the approximation of P^* to the true PF.

Even though GD and IGD take into account the convergence and diversity of the *PFs*, they have some drawbacks. The PF_{true} is needed. When the PF_{true} it is not known, a *best-known* approximated PF can be used. Also, the indicator values are sensitive to the cardinality of the sets. The GD and IGD are not Pareto-compliant according to [Knowles et al., 2006]. The authors presented an empirical evaluation of various quality indicators. They showed that these indicators violate the partial order of weak Pareto dominance. However, they say that it does not mean that they are useless, these indicators can be used to refine the preference structure of a Pareto compliant indicator for approximation sets having identical indicator values.

The **coverage** or **C-metric** (I_{Co}) [Zitzler e Thiele, 1999] measures the "degree" of dominance of a Pareto front over another, i.e., the proportion of the solutions in B that are dominated by at least one solution in A. *C(A,B) = 1* means that all solutions in B are dominated by some solution in A, while *C(A,B)=0* means that no solution in B is dominated by a solution in A.

$$C(A, B) = \frac{|\{u \in B | \exists v \in A : v \; dominates \; u\}|}{|B|} \tag{2.15}$$

where $C(A, B)$ is not necessarily equal to $1 - C(B, A)$. The indicator focuses on the convergence of the sets, and it is not Pareto compliant. Therefore, this metric has been acceptable only to complement the results regarding other indicators, such as the I_{HV}.

2.7.2 Statistical tests

The empirical studies are made by running the algorithms several times on the same problem instance, and obtaining a sample of approximated PFs. The sampling of approximated *PFs* are, usually, represented by their respective quality indicator. Furthermore, a transformation or a normalization on the outcomes is performed before to proceed with the statistical analysis. The purpose of generating samples and defining them as qualitative values is to allow us to describe and make inferences about the underlying stochastic approximation set distributions of (two or more) optimization algorithms, and thus enabling us to compare their performances [Zitzler et al., 2008].

The scientific community has been used different non-parametric statistical tests to evaluate optimization algorithms. In a statistical analysis, hypothesis testing can be employed to draw inferences about one or more populations from given samples. In order to do that, two hypotheses are defined. The null hypothesis H_0 which is a statement of no effect or no difference (i.e., all the samples are assumed to be from the same population), whereas the alternative hypothesis H_1 represents the presence of a difference between the samples. In a statistical test, the term *significant difference* means that a level of significance α is applied to reject the hypothesis H_0 [Derrac et al., 2011].

Friedman's test is used to carry out multiple comparisons among all methods. The null hypothesis for Friedman's test can state equality of medians between the populations. If the Friedman's test rejects the null hypothesis, we can proceed with a *post-hoc* test in order to find the concrete pairwise comparison which has produced differences, and thus rank the algorithms (from the worst to the best) according to the results.

Kruskall-Wallis [Sheskin, 2003] test is another useful non-parametric statistical test used to determine if two samples are from the same population, which is an alternative to the one-way independent-samples Analysis of Variance (ANOVA). In other words, the Kruskall-Wallis tests the null hypothesis and returns the its p-value. This test is well used when no knowledge of the type of distribution is known. If Kruskall-Wallis reject the null hypothesis, a *post-hoc* test, such as *Nemeny*, is used to find the concrete pairwise comparison.

2.8 Conclusions

In this chapter, we have introduced the well-known metaheuristics used for solving MCOPs, including the single-solution methods and the population-based methods. We have also described the general MOEA/D framework which servers as an key ingredient for this thesis.

The complex nature of each MCOP is stated regarding its particular characteristics, such as (i) scalable evaluation function cost, (ii) number of objectives (e.g., related to the number of Pareto optimal solutions, and the objective functions correlation), (iii) fitness landscape features (e.g., number of local optima, deception, dependence between the variables), etc. Moreover, the scientific community have been proposed benchmarks to represent these difficult properties in MCOPs. The benchmarks have been valuable tools to evaluate metaheuristic algorithms before be applied to a real-world instance.

Chapter 3

A decomposition-based binary ACO algorithm for the multi-objective UBQP

3.1 Introduction

In this chapter, we have intended to solve the mUBQP because it is known to be a general model able to represent a broad range of important problems, including those from financial analysis, social psychology, computer aided design, cellular radio channel allocation, traffic management and so on [Zhou et al., 2013b].

Besides, only few metaheuristics have been studied to solve it in the multi-objective case. Due to this, we have proposed a decomposition-based algorithm combined to the metaheuristic Binary ACO to solve mUBQP efficiently. BACO [Fernandes et al., 2007] is suitable to optimize problems with binary representation by building pheromone trails over a $2 \times n$ *dimensional* graph representing the solutions in order to explore/exploit the search space [Fernandes et al., 2007].

The specific goals present in this chapter are: (i) To introduce MOEA/D-BACO as a class of MOEA/D algorithms that incorporate ACO defined on discrete domain; (ii) Empirically analyze the search behavior of the algorithm in different configurations; (iii) To evaluate our proposal against MOEA/D, which uses genetic operators, and against the *best-known* approximated *PFs* from the literature for the set of mUBQP instances considered.

Moreover, some features are incorporated into the MOEA/D-BACO framework to maintain a more diverse population, thus improving its search ability. We have also evaluated our approach with the inclusion of the UBQP-specific local search.

The remainder of this chapter is organized as follows. Section 3.2 describe the mUBQP formulation. Section 3.3 presents the related work. Section 3.4 presents the proposed MOEA/D-BACO and its features. The experiments that empirically show the behavior of the algorithm on a set of mUBQP instances are described in Section 3.6. The conclusions and some trends for future work are presented in Section 3.7.

3.2 Problem Formulation: Multi-objective Unconstrained Binary Quadratic Programming

The UBQP is known to be a general model able to represent a broad range of important problems, including those from financial analysis, social psychology, computer aided design, cellular radio channel allocation, traffic management, and so on. Additionally, some NP-hard problems can be conveniently defined regarding UBQP, such as graph coloring, maxcut, set

packing, set partitioning, maximum clique [Liefooghe et al., 2014, Kochenberger et al., 2014]. The multi-objective UBQP (mUBQP) was proposed by [Liefooghe et al., 2014] as a natural extension of the UBQP. Different MOPs can be formalized regarding mUBQP, e.g., multi-objective knapsack problem, multi-objective maxcut problems, bi-objective coloring problem, etc. [Zhou et al., 2013b].

In the mUBQP, there exists a collection of n items, and each pair of items is associated to $m \geq 2$ profit values that can be positive, negative or zero. Formally, for each objective l, a symmetric $n \times n$ matrix $Q^l = (q^l_{ij})$ is given, and a binary vector $\mathbf{x} = \{x_1, ..., x_i, ..., x_n\}$, $x_i \in \{0, 1\}$ is searched, such that the l objective functions are maximized:

$$max\ F(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} q^l_{ij} x_i x_j;\ l \in \{1, ..., m\}; \qquad (3.1)$$
$$s.t.\ x_i, x_j \in \{0, 1\};\ i, j \in \{1, ..., n\}$$

where $F(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_m(\mathbf{x}))$ is a vector of objective functions. In this case, if $x_i = 1$ and $x_j = 1$, q^l_{ij} is computed.

Liefooghe et al. [Liefooghe et al., 2014] proposed a mUBQP test instance generator which is available at the mUBQP repository[1]. In this case, the instances vary according to (i) the problem size n; (ii) the number of objectives; (iii) a fixed proportion of non-zero values in the matrix (d); (iv) and the objectives correlation strength (ρ) which determines the level of conflict between the objectives. Each value q^l_{ij} is randomly generated according to a uniform distribution in $[-100, 100]$ and the correspondent correlation level ρ. Thus, the values q_{ij} for each objective l can be more correlated or more conflicted. A positive (respectively negative) ρ value decreases (respectively increases) the degree of conflict between the correspondents $\{q^1_{ij}, ..., q^\}_{ij} m$. Thus, the primary difficulties related to the mUBQP are their large problem scale (e.g., $n \geqslant 1000$), and their different objective correlation strengths.

3.3 Related Work

Liefooghe et al. [Liefooghe et al., 2014] proposed a hybrid metaheuristic, called HM, which combines an elitist multi-objective evolutionary algorithm and a single-objective tabu search procedure based on a scalarizing function. In the same paper, the authors proposed a model to generate mUBQP instances. The instances vary according to (i) the number of objectives, (ii) number of variables, (iii) the density (the expected proportion of non-zero entries in the matrix, and (iv) the correlation strength between the objectives. The authors evaluated their algorithm with and without the tabu search and the scalarizing function. They also compared against NSGA-II [Deb et al., 2002] and achieved significantly better results. As we will show latter, we have considered this benchmark to evaluate our proposal.

The authors in [Zhou et al., 2013b] proposed a directional-biased tabu search (DTS) algorithm to generate high-quality solutions using the information of the extreme solutions. First, the DTS optimizes the problem for each objective function in order to obtain extreme solutions. If the extreme solutions for one objective function can not be further improved, then the search gradually changes the direction and optimizes the problem using a series of scalarizing functions. The results showed that the DTS outperforms HM for the same mUBQP instances considered. Further, the best-known results were provided on-line at the mUBQP repository.

[1] http://mocobench.sourceforge.net/index.php?n=Problem.MUBQP

Liefooghe et al. [Liefooghe et al., 2015] focused to solve the bi-objective UBQP. The authors designed and analyzed several approaches using different local search procedures as the main ingredient. The best local search iteratively improves a set of solutions based on a neighborhood structure and the *Pareto* dominance relation. They have achieved better results than the *best-known* approximated *PFs* from the literature.

Overall, these approaches for solving the mUBQP are UBQP-specific because their primary ingredient is a tailored local search called fast incremental local search [Glover e Hao, 2010]. The fast incremental local search takes advantage of the objective function structure of the UBQP to perform a huge number of evaluations based on one-bit-flit move in linear time. It means that, in the fast incremental LS, the neighbor solutions is evaluated without to proceed with the complete evaluation function. This strategy is very useful for UBQP instances with a large scale that have expensive objective functions. However, the main shortcoming of these approaches is the lack of generalization, i.e., if a target problem can not be evaluated n times in linear time, the approach involves a high computational cost.

ACO represents the information learned about the problem as a *pheromone* matrix that measures how likely each component solution is present in a promising solution. Also, ACO allows the incorporation of problem specific knowledge that guides the search [López-Ibánez e Stutzle, 2012, Ke et al., 2013]. In [Ke et al., 2013], the decomposition-based ACO (MOEA/D-ACO) was proposed for solving the multi-objective traveling salesmen problem and the multi-objective knapsack problem. The authors reported significant results compared to MOEA/D using genetic operators. However, they also reported that without the heuristic information the approach performs much poorer than MOEA/D. MOEA/D-ACO is not suitable for the large-scale mUBQP instances, because the size of the pheromone matrix grows exponentially along with the problem size.

Like ACO algorithms , BACO [Fernandes et al., 2007] uses a positive feedback mechanism from the best solutions found so far to maintain pheromone trails represented as a $2 \times n$ graph. The main difference between ACO and BACO is the strategy used to represent and provide the search space in order to explore/exploit it. BACO algorithms has already been applied successfully to discrete and continuous problems [Kong e Tian, 2006, Hu et al., 2011, Fernandes et al., 2007].

3.4 The main algorithm framework

In this section, we present the multi-objective optimization algorithm based on decomposition and BACO metaheuristic for solving the mUBQP. The main difference between our approach and the previous approaches (related work) for solving this problem is that the UBQP-specific local search is not the primary ingredient. The framework is able to control the exploration/exploitation to avoid the premature convergence of the model and maintain a more diverse population.

Combining the ACO paradigm and the binary encoding of the problems, the Binary ACO is represented by a matrix $2 \times n$, which maintains the learned information as pheromone matrices τ. Figure3.1 represents the pheromone matrix for n variables where the arcs $(\tau_{j,0})$ and $(\tau_{j,1})$ represent the pheromones amount of each binary value $\{0, 1\}$. Each path maintains a probabilistic information from the best solutions found so far.

In our framework, a pheromone matrix is maintained for each scalar subproblem i using the set of closest neighbors as the selected population. Let τ^i and p^i be a pheromone matrix and a probabilistic rule, respectively, for each subproblem i. Algorithm 3.1 presents the pseudo-code for the proposed framework.

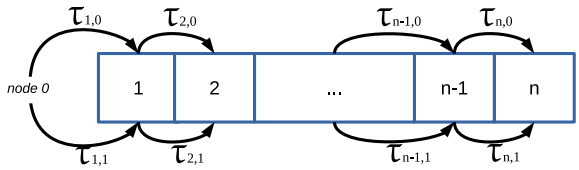


Figure 3.1: The BACO search space for n variables

Algorithm 3.1: MOEA/D-BACO

```
Initialize N uniformly distributed weight vectors and generate N initial permutation
  solutions Pop = {x^1, ..., x^N} randomly and compute every F(x^i).
Initialize EP with the non-dominated solutions from Pop
while a termination condition is not met do
    for each subproblem i ∈ 1, ..., N at each generation do
        Update the pheromone matrices τ^i using solutions x^i ∈ B(i) and their
          corresponding g(x^i|λ^i)
        Construct a solution y using the probabilistic rules associated to p^i
        Compute F(y)
        Update_Neighborhood(B(i), y);
    end for
    Update_EP(Pop, EP)
end while
Return Pop, EP
```

1) Initialization: The N uniform distributed weight vectors $\lambda^1, .., \lambda^N$ are set as described in Section 2.6. The initial population $Pop = \mathbf{x}^1, ..., \mathbf{x}^N$ is generated randomly. Next, their corresponding fitness functions $F(\mathbf{x}^1), ..., F(\mathbf{x}^N)$ are computed. The EP is initialized with the non-dominated solutions from the initial population Pop.

2) Pheromone Matrices Update:

Let b denote a binary value $b = \{0, 1\}$. Each position $\tau^i_{j,b}$ is initialized with a predefined value τ_{start}. The pheromone matrix τ^i, that corresponds to the index subproblem i, is updated according to its solutions neighbors $k \in B(i)$ and their respective fitness scalar value $g(\mathbf{x}^k|\lambda^k)$.

Let Π^i be the set of the solutions that: (i) correspond to the solutions in $B(i)$ and (ii) were just added to EP in the previous generation. The idea is to use the information of the promising solutions to update τ^i. Each position $\tau^i_{j,b}$ is updated as follows.

For each solution $\mathbf{x}' \in \Pi^i$:

$$\tau^i_{j,b} = (1 - \alpha).\tau^i_{j,b} + \alpha.\Delta\tau^i_{j,b} \tag{3.2}$$

where

$$\Delta\tau^i_{j,b} = \sum_{x'_{j=b} \in \Pi^i} \frac{1}{g^{ws}(z^*|\lambda^i) - g^{ws}(\mathbf{x}'|\lambda^i)} \tag{3.3}$$

where α is the evaporation rate (or update rate) and the reference point is $z^*_l = \max\{f_l(\mathbf{x})|\mathbf{x} \in \Omega\}$. The $\Delta\tau^i_{j,b}$ increases according to the frequency of binary values b in x'_j of the solutions in Π^i and its *Weighted Sum* scalar fitness value.

The pheromone matrices store some statistical information of good solutions found so far. Further, the user-specified parameters τ_{max} and τ_{min} control the upper and lower bound

pheromone matrices respectively to avoid premature convergence. If $\tau^i_{j,b} < \tau_{min}$, the value is reset ($\tau^i_{j,b} = \tau_{start}$). If $\tau^i_{j,b} > \tau_{max}$, then $\tau^i_{j,b} = \tau_{max}$.

3) Solution Construction: A new solution is constructed by choosing each position y_j using the following probabilistic rule and the roulette wheel selection mechanism:

$$p^i_{j,b} = \frac{\tau^i_{j,b}}{\tau^i_{j,0} + \tau^i_{j,1}} \tag{3.4}$$

$$p^i_{j,1} = 1 - p^i_{j,0} \tag{3.5}$$

where $p^i_{j,0}$ and $p^i_{j,1}$ are the probability of selecting "0" and "1" for the variable j, respectively.

4) Update Neighborhood: The new sampled solution $\mathbf{y}$ is used to update the current solutions $\mathbf{x}^k \in B(i)$. For each index $k \in B(i)$, if $g(\mathbf{y}\,|\lambda^k) \leq g(\mathbf{x}^k|\lambda^k)$, then $\mathbf{x}^k = \mathbf{y}$. A new solution can update a maximum of n_r current solutions. This strategy prevents one solution having many copies present in the population [Li e Zhang, 2009]. Besides, different strategies of update can be used, such as the global update, where $\mathbf{y}$ can update any solution in Pop.

Afterward, the EP is updated. The procedure adds $\mathbf{y}$ to EP if no solution dominates it and removes the solutions dominated by $\mathbf{y}$ from EP.

3.5 Features of the MOEA/D-BACO

Some particular components are incorporated into the framework, and they are explained as follows.

1) Setting the neighborhood size based on a constant Euclidean distance:

The selection mechanism has a direct impact on the MOEA/D search behavior. In the conventional MOEA/D, all the subproblems have the same neighborhood size T. However, even each λ^i has a different search direction in the objective space, when λ^i and λ^k emphasize the same objective (i.e., an extreme region of the *objective space*), they may have the same set of closest neighbors.

For example: Let $1, 2 \in N$ be the indexes for the weight vectors $\lambda^1 = (1.0, 0.0)$, $\lambda^2 = (0.9, 0.1)$ and the neighborhood size $T = 5$, then, according to the Euclidean distance, the sets $B(1)$ and $B(2)$ could be $B(1) = (\lambda^1, \lambda^2, \lambda^3, \lambda^4, \lambda^5)$ and $B(2) = (\lambda^2, \lambda^1, \lambda^3, \lambda^4, \lambda^5)$, i.e., the subproblems 1 and 2 have the same closest neighbors $B(1) = B(2)$. Thus, when most of (or all) the neighbors solutions in $B(i)$ are located in the same search direction of the *objective space*, the subproblem i is called an extreme subproblem. This selection mechanism has an adverse effect for finding solutions that cover the extreme regions of the objective space because it biases the search towards the central area of the PF.

To overcome this situation, the authors in [Sato, 2015], have introduced a variated neighborhood size to set each $B(i)$. In this mechanism, the neighbors of subproblem i are all the weight vectors having Euclidean distances equal to or smaller than a parameter σ. In this case, the extreme subproblems have smaller T than the subproblem in the middle of the space, since the middle subproblems have more neighbor solutions spread in the *objective space*. The parameter is calculated by:

$$\sigma = \sqrt{2}.\frac{h}{H} \tag{3.6}$$

where h is a user-specified parameter to control the distance σ, and H is the conventional decomposition parameter from [Zhang e Li, 2007] to set the number of subproblems N.

2) Optimal mutation rate for the MOEA/D-BACO:

Sometimes, only the evaporation mechanism is not enough to avoid premature convergence. To prevent this issue, we have incorporated a stochastic mutation to promote diversity. As in [Mahnig e Mühlenbein, 2001], we use the Bayesian *priors* as an effective approach to introduce a mutation-like effect into the MOEA/D-BACO.

The conventional computation of the univariate probability of a variable x_j can be denoted as $p(x_j) = \frac{freq}{S}$, where $freq$ is the frequency, e.g., the proportion of "1s" in the j^{th} variable from the selected population S. The Bayesian approach assumes that the probability of "1s" is computed as $p(x_j) = \frac{freq+r}{S+2r}$, where the *hyperparameter r* has to be chosen in advance. To relate the Bayesian *prior* to the mutation rate (μ) of each variable used by the genetic operators, the authors used the following theorem: *For binary variables, the expectation value for the probability using a Bayesian prior with parameter r is the same as mutation with mutation rate* $\mu = \frac{r}{S+2r}$ *and using the maximum likelihood estimate* [Mahnig e Mühlenbein, 2001].

Therefore, to introduce the Bayesian *priors* into our approach, the pheromone amount $\Delta\tau^i_{j,b}$ is assigned as the following equation:

$$\Delta\tau^i_{j,b} = \frac{\Delta\tau^i_{j,b} \,.\, T^i + r}{T^i + 2r} \tag{3.7}$$

where T^i is the size of the set Π^i from equation (5).

3) Diversity preserving sampling:

One of the shortcomings of some evolutionary algorithms is the lack of diversity in the population due to the generation of solutions already present in the current population, which can lead to premature convergence. As a course to avoid this situation, we added a simple procedure in which each new constructed solution is tested for presence in the neighborhood. If $\mathbf{y} \in B(i)$, then the solution is discarded and a new solution is sampled until $\mathbf{y} \notin B(i)$ or a number of trials is reached.

3.6 Experiments

In this section, we firstly analyze the impact of the enhancements incorporated into the algorithm. Next, we evaluate the algorithm search behavior regarding the learning rate parameter "α". Finally, an experimental study compares the algorithm to the MOEA/D and the *best-known* approximated *PF* available.

Liefooghe et al. [Liefooghe et al., 2014] proposed a mUBQP test instance generator, which is available at the mUBQP repository[2]. Likewise, the repository contains a number of test instances already generated and their respective approximated *PFs* achieved by merging the outcomes from different approaches and executions.

We have used the same instance configurations used by the previous work [Liefooghe et al., 2014, Zhou et al., 2013b, Liefooghe et al., 2015]. Thus, the instances vary according to: (i) the number of objectives $m = \{2, 3\}$, (ii) objective correlation strength $\rho = \{-0.5, -0, 2, 0.0, 0.2, 0.5\}$ (from the most conflicting degree to the less conflicting), and (iii) number of variables $n = \{1000\}$. The density of non-zero values in the cell of the matrix of the UBQP is $d = 0.8$. We have also included instances with $n = \{500, 750\}$ that we have generated using the mUBQP instance generator.

[2] http://mocobench.sourceforge.net/index.php?n=Problem.MUBQP

3.6.1 Parameters Setting

We have executed 30 independent runs for each algorithm configuration for each test instance. The following parameters setting were used in the experimental studies:

1. *Number of subproblems* (N)*:* According to the formulation $C_{H+m-10}^{m-1} = N$ [Zhang e Li, 2007], we set $H = \{200, 9\}$, and consequently $N = \{201, 220\}$ for two and three objectives respectively.

2. *Neighborhood sizes*: The parameter h (from equation (3.6)) is set to generate neighborhoods whose size is no larger than $T = 20$. Thus, for $m = 2$ and $m = 3$, we have $h = 0.04$ and $h = 0.132$, respectively.

3. *Maximal number of replacements by a new solution* (n_r)*:* As in [Li e Zhang, 2009], a new sampled solution can replace a maximum of $n_r = 2$ solutions.

4. *Scalarization function:* We have applied both the *Weighted Sum* and *Tchebycheff* approach. As both have achieved very similar results, we only present the results using the *Tchebycheff*. The reference point $z_l^* = \max\{f_l(\mathbf{x})|\mathbf{x} \in \Omega\}$ is always updated after the algorithm generates a new solution.

5. *Pheromone initialization and bounds:* According to a preliminary study, we set $\tau_{start} = 0.5$, $\tau_{min} = 0.001$ and $\tau_{max} = 1$.

6. *Stopping condition:* The algorithms stop after n generations, i.e., for each test instance, the number of generations is set according to its problem-size n. Consequently, the correspondent total number of fitness evaluations is $N \times n$.

3.6.2 Impact of the MOEA/D-BACO components

Each feature has been analyzed separately, and all of them together. Figure 3.2 represents the evaluation of 5 different MOEA/D-BACO configurations. Each configuration is defined in the following:

1. “standard": represents the algorithm with no additional feature.

2. “+cd": represents the algorithm with the variated neighborhood size.

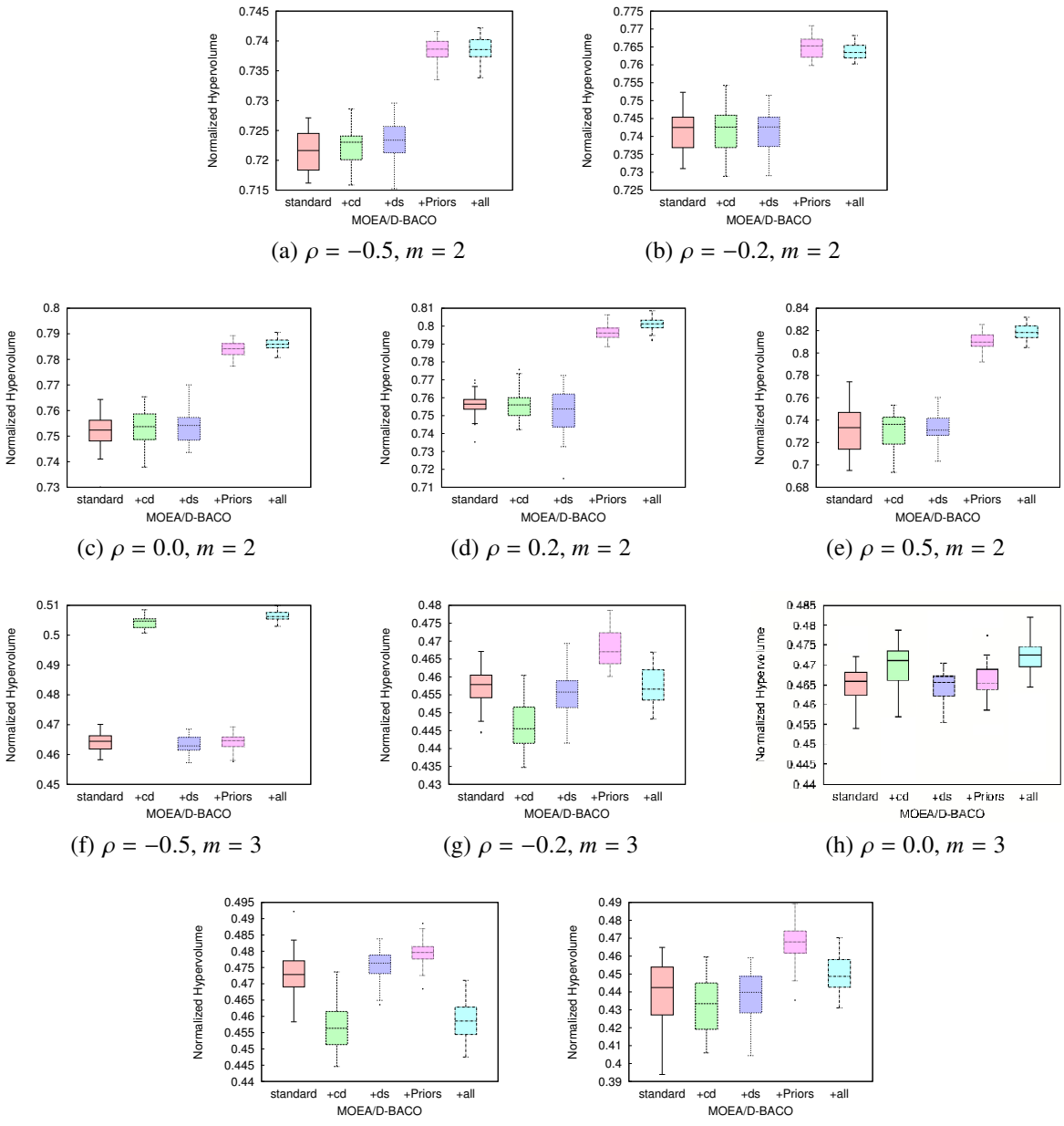
3. “+ds": is the algorithm with the diversity preserving procedure.

4. “+Priors": represents the algorithm with the Bayesian *Priors* mutation-like effect.

5. “+all": is the algorithm using the three features together.

In Figure 3.2, each subfigure represents the values of the normalized *HV* measure obtained by 30 runs for each MOEA/D-BACO configuration. We applied the statistical test *Kruskall-Wallis* [Derrac et al., 2011] (5% of significance level) on the results obtained by the *HV* to check if there is a significant difference between the five configurations.

According to these results, we make the following remarks:



(a) $\rho = -0.5$, $m = 2$ (b) $\rho = -0.2$, $m = 2$

(c) $\rho = 0.0$, $m = 2$ (d) $\rho = 0.2$, $m = 2$ (e) $\rho = 0.5$, $m = 2$

(f) $\rho = -0.5$, $m = 3$ (g) $\rho = -0.2$, $m = 3$ (h) $\rho = 0.0$, $m = 3$

(i) $\rho = 0.2$, $m = 3$ (j) $\rho = 0.5$, $m = 3$

Figure 3.2: Means plot for the normalized *HV* indicator for several configurations of MOEA/D-BACO for 1000 variables

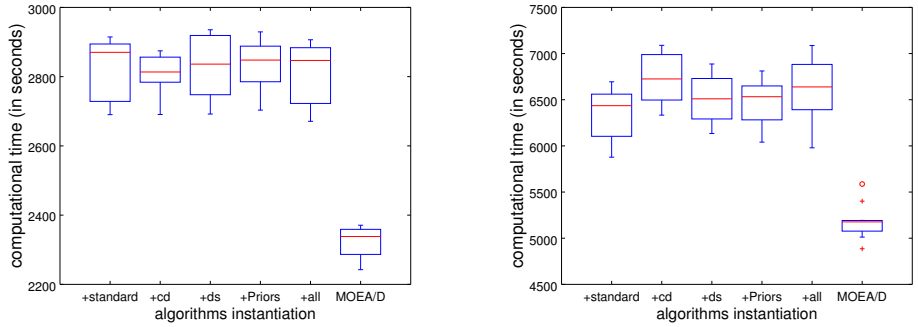
- The variated neighborhood size (+cd) has no significant difference in comparison to the (+standard) for the test instances with two objectives. However, it has a negative impact for the test instances with three objectives, except for the instance with the highest degree of conflict between the objectives $\rho = -0.5$. The reason is that the (+cd) enhancement generates more diverse neighborhood sizes as the number of the objectives increases due to a large number of weight vectors λ^i with the same Euclidean distance, and it helps to find solutions in the extreme regions of the objective space.

- The diversity preserving sampling (+ds) has not achieved significant difference against the +standard. It means that for the mUBQP instances evaluated, the probability of sampling a solution already in the population is very low.

- The Bayesian *Priors* (+Priors) has a positive impact in the most of the test instances, which confirms that the mutation-like effect is essential for the MOEA/D-BACO to avoid the premature converge of the model.

Overall, the enhancements have distinct behaviors for the different instance configurations. However, all the components together have achieved the best results in most of the cases. We have not considered a self-adapted approach for choosing the best algorithm setting per instance configuration. Thus, we have decide to use the configuration (+all) for the next experiments.

Moreover, Figure 3.3 presents the computational cost (in seconds) of each algorithm configuration for a representative test instance with two and three objectives. The results show that the different enhancements do not have a significant impact on the computational cost of MOEA/D-BACO.



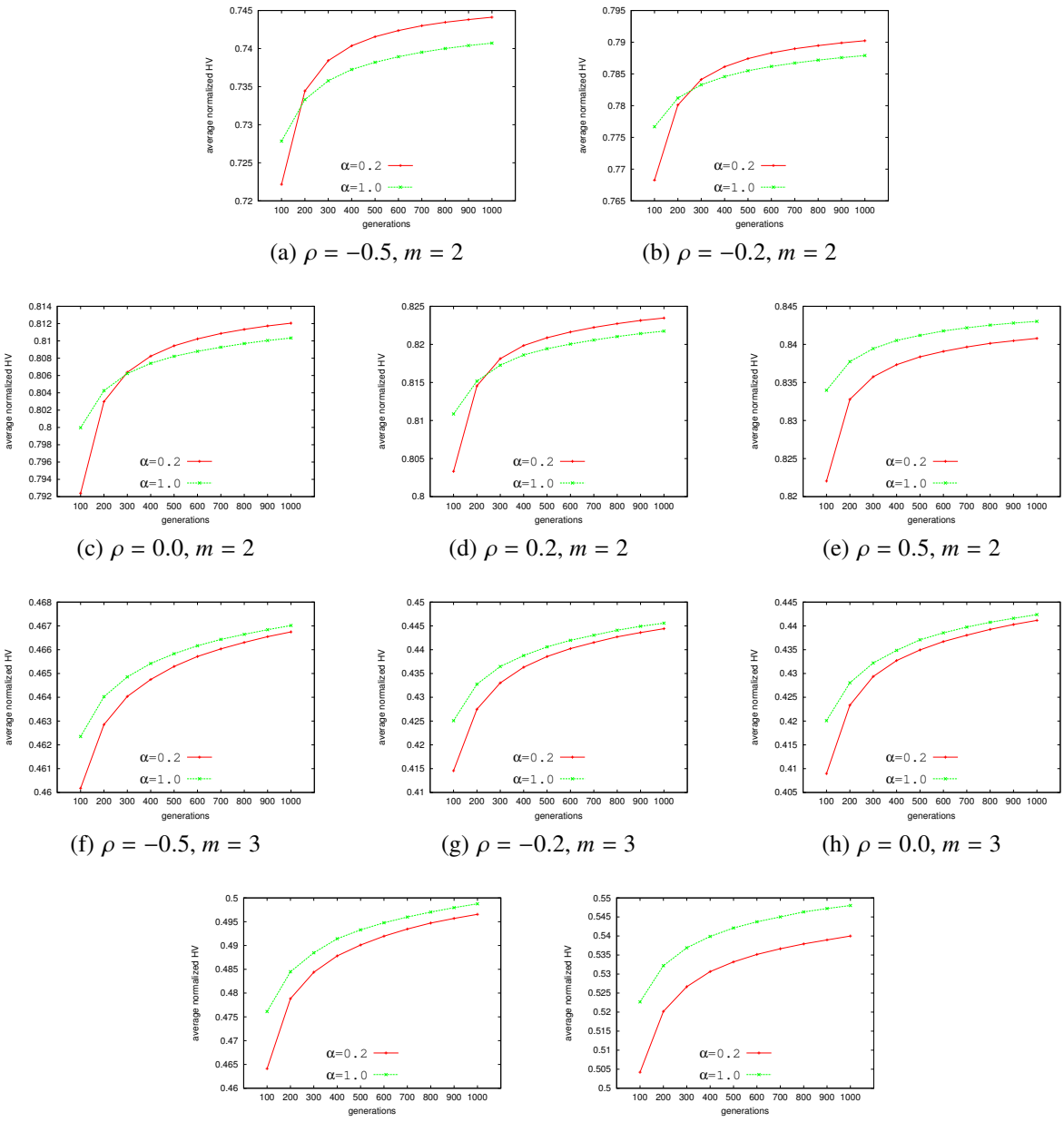
(a) $\rho = -0.5$, $m = 2$ (b) $\rho = -0.5$, $m = 3$

Figure 3.3: Means plot for computational time in seconds for 1000 variables, the correlation strength $\rho = -0.5$ and two and three objectives

3.6.3 Impact of the update rate α

This section presents the search behavior of the MOEA/D-BACO throughout the generations. The update rate α has a direct impact on the pheromone matrices update. This parameter deals with the convergence of the model. A small α value means that the algorithm will need a greater number of generations to converge. When $\alpha = 1.0$ the model uses only the new increment $\Delta\tau^i_{j,b}$ to update $\tau^i_{j,b}$ (equation 3.2).

Figure 3.4 shows the average of the normalized *HV* achieved by 30 runs throughout the generations with $\alpha = \{0.2, 1.0\}$ for the test instances with 1000 variables, two and three objectives and the different objectives correlation strength. According to the results, in the first generations (e.g., 100 generations), MOEA/D-BACO with $\alpha = 1.0$ converges faster than with $\alpha = 0.2$. However, for $m = 2$ and all objectives correlations strength, except for the weakest conflicting $\rho = 0.5$, the algorithm with $\alpha = 0.2$ has achieved better *HV* results after 1000 generations. It shows that a low α value contributes to avoid the premature convergence of the pheromone matrices, thus achieving better *HV* results after a number of generations. For $m = 3$, the algorithm with $\alpha = 1.0$ has produced better results. It indicates that the complexity of the problem increases with three objectives, and thus a higher number of generations is needed to achieve better quality solutions.



(a) $\rho = -0.5$, $m = 2$ (b) $\rho = -0.2$, $m = 2$

(c) $\rho = 0.0$, $m = 2$ (d) $\rho = 0.2$, $m = 2$ (e) $\rho = 0.5$, $m = 2$

(f) $\rho = -0.5$, $m = 3$ (g) $\rho = -0.2$, $m = 3$ (h) $\rho = 0.0$, $m = 3$

(i) $\rho = 0.2$, $m = 3$ (j) $\rho = 0.5$, $m = 3$

Figure 3.4: Average normalized *HV* values from MOEA/D-BACO with $\alpha = \{0.2, 1.0\}$ throughout the generations for the mUBQP instances $n = 1000$

3.6.4 Comparison to the MOEA/D

In this section we compare MOEA/BACO to MOEA/D using the *uniform crossover* and the mutation rate $\mu = \frac{1}{n}$. According to the results in the previous section, we set $\alpha = 0.2$.

Table 3.1 summarizes the average of the normalized *HV* results of the final approximated *PF* obtained by each instantiated algorithm over the 30 runs for each test instance. If the algorithms have a significant difference according to the Kruskall-Wallis test at 5% of significance level, the best-ranked results are highlighted in boldface. Table 3.2 presents the coverage indicator between MOEA/D-BACO and MOEA/D over the 30 runs for each benchmark configuration.

We make the following remarks:

Table 3.1: Average *HV* values of the final approximated *PF* obtained. When the algorithms have statistical differences, the best results are highlighted in bold face.

Instance		$m = 2$		$m = 3$	
n	ρ	MOEA/D	MOEA/D-BACO	MOEA/D	MOEA/D-BACO
500	-0.5	0.7499	0.7495	0.5275	**0.5322**
	-0.2	0.7544	**0.7604**	0.5132	**0.5225**
	0.0	0.8011	0.7998	**0.4954**	0.4850
	0.2	0.8135	0.8151	**0.4486**	0.4339
	0.5	0.7650	**0.7811**	**0.5492**	0.5396
750	-0.5	0.7373	0.7377	0.4987	**0.5004**
	-0.2	0.7714	**0.7825**	0.5105	**0.5216**
	0.0	0.7722	**0.7768**	0.4799	0.4721
	0.2	0.7948	**0.7995**	0.5023	0.4930
	0.5	0.8090	**0.8280**	**0.5486**	0.5035
1000	-0.5	0.7304	**0.7356**	0.5154	**0.5177**
	-0.2	0.7694	**0.7766**	0.4686	**0.4695**
	0.0	0.7840	**0.7916**	0.4521	0.4529
	0.2	0.7962	0.8024	0.4734	0.4698
	0.5	0.8000	**0.8145**	**0.4912**	0.4642

Table 3.2: *C-metric* values [0, 1] between MOEA/D-BACO (A) and MOEA/D (B)

Instance		$m = 2$		$m = 3$	
n	ρ	C(A,B)	C(B,A)	C(A,B)	C(B,A)
500	-0.5	0.1291	**0.852**	**0.4185**	0.3455
	-0.2	0.3128	**0.619**	**0.6869**	0.1632
	0.0	0.06081	**0.9121**	0.13	**0.7403**
	0.2	0.08736	**0.8333**	0.07454	**0.7914**
	0.5	0.2691	**0.6964**	0.2162	**0.6526**
750	-0.5	**0.5581**	0.4471	**0.4054**	0.3883
	-0.2	**0.7511**	0.2262	**0.4401**	0.6483
	0.0	0.2978	**0.694**	0.1492	**0.7069**
	0.2	0.3384	**0.6828**	0.06919	**0.812**
	0.5	**0.6548**	0.2742	0.001951	**0.9765**
1000	-0.5	**0.7169**	0.2599	**0.6821**	0.3512
	-0.2	**0.7356**	0.5263	**0.5831**	0.2313
	0.0	**0.6996**	0.4586	0.3761	**0.6712**
	0.2	**0.5276**	0.3829	0.192	**0.6336**
	0.5	**0.7632**	0.25	0.04287	**0.9095**

- Table 3.1 shows that, regarding *HV* indicator, the MOEA/D-BACO outperforms MOEA/D, with a significant difference, in 16 of the 30 instances, and did not achieve statistical difference in 9 cases. For $m = 2$, MOEA/D-BACO has produced the best results. For $m = 3$, the difference between the algorithms is higher when the objectives are strongly conflicting (e.g., $\rho = \{-0.5, -0.2\}$);

- Regarding the *C-metric* indicator, Tables 3.2 shows that, for $m = 2$, MOEA/D-BACO achieved better results as problem-size increases. For $m = 3$, the proposed algorithm dominates a larger percentage of solutions from MOEA/D when the objectives are strongly conflicting ($\rho = \{-0.5, -0.2\}$), in accordance to the results with the *HV* indicator;

- In general, the results from Table 3.1 and 3.2 show that, regarding the objective correlation strength, MOEA/D-BACO significantly outperforms MOEA/D mainly when the objectives are strongly conflicting. Also, as reported in Figure 3.3, using the same number of generations, MOEA/D-BACO has a higher computation cost than MOEA/D.

3.6.5 Hybridization with the problem-specific local search

Since the fast incremental local search [Glover et al., 1998, Glover e Hao, 2010] has been the main ingredient of the approaches for solving the UBQP and the mUBQP, we have also hybridized our algorithm with it to evaluate its impact on the final results. In the following, we describe how we have incorporated the fast incremental UBQP local search in our framework.

After each single-objective scalar subproblem i generates a new solution $\mathbf{y}$, a simple iterated random local search is applied. For $n/10$ times, a random position (variable) is chosen and this binary value is flipped generating $\mathbf{y}'$, and then the procedure calculates its *move gain*. If $g(\mathbf{y}'|\lambda^i) > g(\mathbf{y}|\lambda^i)$, then $\mathbf{y} = \mathbf{y}'$.

Due to the drastic computational difference between the complete evaluation function cost (Eq. 3.1) and the fast incremental evaluation function cost, we have evaluated the algorithms according to two stop conditions: (i) the previous fixed number of generations, and (ii) using a limited computational cost. The computational cost used was $n \times m \times 300$ milliseconds. Table 3.3 and Table 3.4 present these results respectively according to the *HV* indicator.

Table 3.3: Average *HV* values of the final approximated *PF* obtained by MOEA/D and MOEA/D-BACO with and without the fast incremental local search procedure after n generations.

Instance		m=2			
n	ρ	MOEA/D	MOEA/D-BACO	MOEA/D+ls	MOEA/D-BACO+ls
	-0.5	0.7345	0.7395	0.7468	**0.7473**
	-0.2	0.7678	0.7747	0.7831	**0.7866**
	0.0	0.7874	0.7946	**0.8064**	**0.8030**
	0.2	0.7945	0.8004	**0.8231**	0.8172
	0.5	0.7921	0.8060	0.8256	**0.8327**
1000				m=3	
	-0.5	0.4326	0.4345	0.4508	**0.4595**
	-0.2	**0.4237**	**0.4243**	0.3968	0.4170
	0.0	0.4480	**0.4503**	0.4230	0.4407
	0.2	**0.4813**	0.4786	0.4595	0.4617
	0.5	0.5051	0.4823	**0.5160**	0.4801

Table 3.4: Average *HV* values of the final approximated *PF* obtained by MOEA/D and MOEA/D-BACO with and without the fast incremental local search procedure after $n \times m \times 300$ milliseconds.

Instance		m=2			
n	ρ	MOEA/D	MOEA/D-BACO	MOEA/D+ls	MOEA/D-BACO+ls
	-0.5	0.6726	0.6160	**0.7208**	0.6948
	-0.2	0.6882	0.6139	**0.7609**	0.7271
	0.0	0.6997	0.5954	**0.7862**	0.7504
	0.2	0.7073	0.5353	**0.8140**	0.7715
	0.5	0.6859	0.4631	**0.8531**	0.8011
1000				m=3	
	-0.5	0.3330	0.3110	**0.3989**	0.3713
	-0.2	0.3279	0.2111	**0.3459**	0.3233
	0.0	0.3395	0.2199	**0.3513**	0.3239
	0.2	0.3849	0.2047	**0.4092**	0.3617
	0.5	0.4451	0.1269	**0.5680**	0.4515

Regarding the fixed number of generations, the results from Table 3.3 show that the algorithms with the local search have achieved the best results with a significant difference for

$m = 2$. However, for $m = 3$, the local search has not achieved a significant impact on the results. In some cases, the algorithms without the local search achieved the best results. It means that, for 3 objectives, the exploitation performed by the local search can have a detrimental effect, i.e., it avoids the diversity in the objective space.

Regarding the computational cost, the results from Table 3.4 show that MOEA/D has less computational cost than MOEA/D-BACO. Furthermore, MOEA/D+ls have achieved the best results. However, as it is shown in Table 3.3, even though our approach is slower, it is able to "continue evolving", and thus obtain better results.

3.6.6 Comparison to the best-known referent *PFs*

The *best-known* approximated *PFs*, which are available at the mUBQP repository, were generated by the authors merging the results from the different approaches, executions, and stopping criteria [Zhou et al., 2013b]. Table 3.5 presents the absolute *HV* values obtained by MOEA/D-BACO with and without the local search, and the referent approximated *PFs*.

Table 3.5: *HV* values of the final approximated *PFs* obtained.

Instance		m=2		
n	ρ	MOEA/D-BACO	MOEA/D-BACO+ls	referent *PF*
	-0.5	0.7330	0.7380	**0.7555**
	-0.2	0.7677	0.7733	**0.8035**
	0.0	0.7860	0.7916	**0.8131**
	0.2	0.7738	0.7847	**0.8364**
	0.5	0.8112	0.8205	**0.8455**
1000		m=3		
	-0.5	0.4399	**0.4571**	0.4421
	-0.2	**0.4531**	0.4409	0.4431
	0.0	0.4323	0.4163	**0.5171**
	0.2	0.4374	0.4154	**0.6148**
	0.5	0.4983	0.4915	**0.6251**

The comparison to the *best-known* approximated *PFs* presented in Table 3.5 clearly indicates the competitive results obtained by our approach mainly for three objectives and high conflict correlation strength. These results show that the approaches (that make a more global search) can achieve competitive results compared to the UBQP-specific algorithms. Additionally, we made our *best-known* results available on-line[3] for further investigations.

3.7 Conclusion and Future Work

In the presented MOEA/D-BACO framework, each scalar optimization subproblem maintains a pheromone matrix $n \times 2$. Each subproblem updates its model according to the positive feedback from its best neighbor solutions found so far. We have incorporated enhancements to the MOEA/D-BACO to maintain diversity.

An experimental study was conducted to evaluate the different components of MOEA/D-BACO framework. The results showed that each feature has a different impact depending on the

[3]Available at https://github.com/MuriloZangari/mubqp-best-known

characteristics of the mUBQP instances that vary according to the number of objectives, number of variables and the objectives correlation strength.

We have analyzed the search behavior of MOEA/D-BACO with two update rate α values. The results showed that a low α avoids the fast convergence of the probabilistic models, and consequently it needs a large number of generations to converge to the true *PF*. It is because algorithm leads to a more parsimonious update of the model.

MOEA/D-BACO is slower than MOEA/D. However, using the number of generation as stopping condition, our approach achieves better results regarding *HV* indicator and *C-metric* in most of the cases. In comparison to the *best-known* sets, MOEA/D-BACO achieves competitive results. Moreover, the local search is not the major ingredient of our algorithm, which leads us to have a more general approach which can be evaluated to solve other optimization problems with binary representation.

It is known that ACO algorithms and EDAs, such as PBIL [Baluja, 1994] and UMDA [Pelikan et al., 1999], have a close relation [Blum e Roli, 2003]. These algorithms maintain a model based on distribution of probabilities, and the variables are considered independent. Therefore, we also have investigated the incorporation of this kind of EDAs within our MOEA/D framework. This subject is addressed in the next chapter, where we first discuss and analyze different PBIL's implementations from the literature. We show how these differences affects the search ability of the algorithm. We also present a comparison to MOEA/D-BACO.

Chapter 4

Not all PBILs are the same: Unveiling the different learning mechanisms of PBIL variants

4.1 Introduction

Following with our research on metaheuristics to solve MCOPs, we have started to explore new ideas for it. We have noticed that BACO (presented in the previous chapter) and PBIL [Baluja, 1994] have a close relation because both use machine learning techniques to maintain a model representing the solutions from the *search space*.

In fact, PBIL is one of the simplest model-based evolutionary algorithms and arguably one of the first EDAs [Mühlenbein e Paass, 1996]. The genesis of PBIL was largely influenced by previous work on population-based recombination in GAs and the concept of competitive learning, as applied in neural networks [Hertz, 1991]. In a string of papers, Baluja et al. [Baluja, 1994, Baluja e Caruana, 1995, Baluja, 1997], described how the exchange of information between solutions, implicit in the behavior of crossover operators, could be efficiently transformed in the explicit manipulation of probabilistic vectors describing the statistics of the population.

In simple terms, at each generation, PBIL works by updating a vector describing the univariate statistics of the best solutions. The update of this simple model is governed by a parameter that works as a learning rate. The model is used to generate new candidate solutions and the loop comprising selection, model learning, and model sampling continues until a stop criterion is met.

The simplicity of PBIL and its ability to retain much of the performance of GAs with uniform and one-point crossover operators has contributed to its popularity. In general, some of the most praised characteristics of PBIL are: its easy implementation [Baluja, 1997] in comparison to more complex EDAs, and its robustness [da Silva e Schirru, 2014]. Besides, PBIL has been applied to a variety of real-world problems including energy applications [da Silva e Schirru, 2014, Folly, 2013, Folly, 2014], automatic control [dos Santos Coelho e Grebogi, 2010], biomedical problems [McCall et al., 2008], robotics [Kang et al., 2014], multiobjective complex network problems for community strcture detection [Ma et al., 2016], and other problems [Chaves-González et al., 2008, Chen et al., 2016, Salvá et al., 2013]. Also, extensions for continuous problems have been introduced [Sebag e Ducoulombier, 1998, Yuan e Gallagher, 2003]. Usually, the authors have proposed PBIL algorithms with some enhancements for efficiently solving the target problem (e.g., self-adaptive approach [da Silva e Schirru, 2014], multiple-population PBIL [Folly, 2014], hybrid approaches [dos Santos Coelho e Grebogi, 2010], differ-

ent learning and sampling procedures to guarantee a higher diversity [Ventresca e Tizhoosh, 2008, Ma et al., 2016], parallel schemes [Folly, 2013]).

In most of those works, the authors stated that their PBIL algorithms outperformed the conventional GA approaches [Kang et al., 2014]. However, some works presented that PBIL can be outperformed by EDAs that use more complex models or tuned search strategies [Yang e Yao, 2005, Chaves-González et al., 2008]. Moreover, theoretical analyses of PBIL have been conducted. These papers have focused on the convergence proof of the algorithm and its expected behavior regarding its ability to find local optima of the function [González et al., 2001b, Yang e Yao, 2005, Rastegar e Hariri, 2006, Li et al., 2011].

Notably, and this is the main claim presented in this chapter, the numerous reported applications of the discrete PBIL can be split into two main groups according to the way the learning step of the algorithm is interpreted. The critical aspect of this split is that two essentially different algorithms (in terms of their learning mechanisms) are considered in the current literature as a unique algorithm. Furthermore, for one of these PBIL variants, the algorithm is not well specified, in the sense that the commonly used description of the algorithm allows different implementations.

In this chapter, we show that the differences between the identified PBIL variants are critical in terms of the effect that they can produce in the search process. We conduct an analysis from a theoretical and empirical point of view. We did not find any previous published analysis of the fact the PBIL algorithm has different interpretations regarding its learning mechanism.

The chapter is organized as follows: Section 4.2 discusses the PBIL variants reported in the literature. In Section 4.3, we derive formulas for computing the mean univariate probabilities from the two PBIL variants when all possible orders are considered. Section 4.4 presents the experimental studies using a number of well-known optimization functions. In Section 4.6, we conclude the chapter and discuss topics for future research.

4.2 PBIL

Algorithm 4.1 shows the pseudo-code of the PBIL, adapted from [Baluja, 1994], in which the algorithm was first introduced. Each position of the probabilistic vector is initialized $p(x_j) = 0.5$. Next, the solutions in population Pop are initialized randomly. While a termination condition is not met, the *best* solution is determined according to the quality function, and then the univariate probabilistic vector $p(\mathbf{x})$ is updated (*line 7*). After, it samples a new solution $\mathbf{y}$ and add it to Pop.

One distinguished feature of this PBIL presented in Algorithm 4.1 is that the update of the univariate vector is done using the best solution found in each generation.

It is also acknowledged in [Baluja, 1994] that, when large populations are used, adjusting the prototype vector based upon the single best solution vector in each generation has the potential of ignoring a large amount of the work and exploration performed by the algorithm. The straightforward solution proposed is to move the prototype vector in the direction of the best ($S << Pop$) solutions. An enumeration regarding possible manners to implement this variant is presented in [Baluja, 1994]. One of the suggested ideas is to move the probability vector equally in the direction of each of the selected solutions. An implementation of this idea is presented in [Baluja, 1997].

The main differences in the learning mechanism between the PBIL presented in [Baluja, 1997] and the original algorithm are described in the pseudocode of Algorithm 4.2. Two characteristics of this variant are that the solutions in Pop are first sorted according to the evaluation of the solutions (Step 1), and the probabilistic model learned is sensitive to this order.

Algorithm 4.1: PBIL (adapted from the original formulation [Baluja, 1994])

$p(\mathbf{x}) \leftarrow$ initialize each position of the probability vector $p(x_j) = 0.5 \; \forall j \in 1, ..., n$
$Pop \leftarrow$ Generate N solutions randomly
For each solution $\mathbf{x}$, compute its fitness function $F(x)$
while *a termination condition is not met* **do**
    $best \leftarrow$ the solution corresponding to best fitness
    for *each variable, the corresponding entry probability is updated* **do**
        $p(x_j) = (1.0 - \alpha) * p(x_j) + (\alpha * best_j)$
    end for
    $\mathbf{y} \leftarrow$ sample a new solution using $p(\mathbf{x})$
    For each variable, if (random$(0, 1) < \mu$) $y_j = 1 - y_j$
    $Pop \leftarrow$ add $\mathbf{y}$ to the population
end while

This dependence on the order is due to the fact that the values of the probabilistic vector are iteratively modified within the loop that passes over all the selected solutions. The last solution vector in the order will have a stronger impact on the final configuration of the probability vector. We call this variant the *order-sensitive* PBIL, or in short PBIL-OS.

Algorithm 4.2: PBIL (Order-sensitive variant) learning procedure [Baluja, 1997]

Pop = sort(Pop)
#Update Probability Vector towards best solutions
for $i = 1$ *to NUMBER_OF_SOLUTIONS_TO_UPDATE_FROM (S)* **do**
    for $j = 1$ *to* n **do**
        $p(x_j) = (1.0 - \alpha) * p(x_j) + (\alpha * x_j^i)$
    end for
end for

Finally, another interpretation of PBIL assumes that, before updating the probability vector, a vector $r(\mathbf{x})$ with the univariate probabilities of the S selected solutions is computed. The auxiliary vector $r(\mathbf{x})$ is then used to update the vector of the univariate probabilities. The pseudo-code of this variant is shown in Algorithm 4.3. We call this variant PBIL-iUMDA, because the strategy used to update the probabilities is the same as that proposed for the iUMDA in [Mühlenbein, 1997].

Algorithm 4.3: PBIL (iUMDA variant) learning procedure [Mühlenbein, 1997]

#Compute current probabilities
for $j = 1$ *to* n *do* **do**
    $r(x_j) = \frac{\sum_{i=1}^{S} x_j^i}{S}$
end for
#(Update Probability Vector)
for $j = 1$ *to* n *do* **do**
    $p(x_j) = (1.0 - \alpha) * p(x_j) + \alpha * r(x_j)$
end for

Examples of the original variant include [Baluja, 1994, Baluja, 1997, Chaves-González et al., 2008, Ma et al., 2016]. Order-sensitive examples are those de-

scribed in [Yang e Yao, 2005]. The iUMDA variant is at the core of the following papers [González et al., 2001a, González et al., 2001b, Khan, 2014, Mendiburu et al., 2006].

4.3 Expected values of univariate models for PBIL-OS and the PBIL-iUMDA

In this section, we study the algorithm variants described in the previous section PBIL-OS and PBIL-iUMDA from a theoretical point of view, looking at the equations that are used to update the probabilistic model. Our goal is to determine how the choice of the learning algorithm that defines each PBIL variant influences their expected univariate probabilities.

Theoretical analyses of PBIL have been previously conducted. However, they have primarily focused on the convergence proof of the algorithm. Most of the theoretical analysis of PBIL has addressed the iUMDA variant [Li et al., 2011, Lozano, 2000, Rastegar e Hariri, 2006, Rastegar, 2011] and this analysis usually assumes that the learning rate parameter α is very close to zero [González et al., 2001b, Li et al., 2011, Rastegar e Hariri, 2006]. In [González et al., 2001b], the authors analyze the expected behavior of PBIL for the *One-Max* function. They presented a mathematical proof for the algorithm's convergence behavior, which depends on the initial value of the probabilistic vector $p(\mathbf{x})$ and the value of the learning rate parameter (α). Other papers have proposed proofs of PBIL convergence to local optima [Rastegar e Hariri, 2006].

In order to build some intuition about how the PBIL learning variants use the selected solutions to generate the model, we start by presenting some illustrative examples of the effect that the order of the selected solutions has on the univariate model. Then, for each PBIL variant, we derive the formula of the expected univariate probabilities if all the possible orderings of the selected solutions were considered.

To illustrate the influence that the order of the solutions has, let us consider two populations of four solutions, where all the solutions have five variables, $(x_1, x_2, x_3, x_4, x_5)$.

$$Pop_A = (00000, 00111, 11000, 11111) \tag{4.1}$$

$$Pop_B = (11111, 00000, 00111, 11000) \tag{4.2}$$

Pop_A and Pop_B comprise identical solutions but they are ordered in a different fashion. We will assume that PBIL will learn its probability model from each population. The univariate probability model computes the univariate probability for each variable, but to ease the presentation, we will focus on a single variable. Let us concentrate on the last variable, x_5. For this variable, we can extract its corresponding column vectors v_A and v_B from the two populations. v_A and v_B have the form $(x_5^1, x_5^2, x_5^3, x_5^4)$. $v_A = (0101)$, $v_B = (1010)$.

To learn the univariate probability of this variable the way PBIL-OS does, we apply the updating equation $p(x_j) = (1 - \alpha)p(x_j) + \alpha x_j$ to compute $p_A(x_5)$ and $p_B(x_5)$ using v_A and v_B, respectively. Assuming that the initial marginal probability for the two cases is $p(x_5) = b$, the final values of $p_A(x_5)$ and $p_B(x_5)$ after iterating over the four solutions in the population are:

$$p_A(x_5) = b(1 - \alpha)^4 + \alpha(1 - \alpha)^2 + \alpha \tag{4.3}$$

$$p_B(x_5) = b(1 - \alpha)^4 + \alpha(1 - \alpha)^3 + \alpha(1 - \alpha) \tag{4.4}$$

It is clear that the expression of the univariate probabilities learned from the two populations is different and depends on the order of the solutions. For example, for $b = 0.5$ and

$\alpha = 0.2$, we have $p_A(x_5) = 0.5648$ and $p_B(x_5) = 0.672$, which is a significant difference from the univariate probabilities obtained from the two populations. This is not the usual behavior of methods applied to learn probabilistic models in EDAs. The rule in EDAs is that the probabilistic model does not depend on the order of the solutions in the selected population.

The analysis presented for the variable x_5 can be generalized to any variable i and any order of the solutions in the population. Equation 4.5 shows the parameterized expression of $p(x_j)$ for any order of a population of 4 individuals.

$$p(x_j) = b(1-\alpha)^4 + x_i^1\alpha(1-\alpha)^3 + x_i^2\alpha(1-\alpha)^2 + x_i^3\alpha(1-\alpha) + x_i^4\alpha \tag{4.5}$$

For PBIL-iUMDA, the result of the updating rule for any order will be the same. For example, assuming a selected population of size $|S|$ with exactly k solutions with value 1 for variable x_i^j, and $|S| - k$ solutions with value 0, then $r_i = \frac{|S|}{N}$ and the result of the application of the update rule would be $p(x_j) = b(1-\alpha) + \frac{\alpha k}{|S|}$.

The computation of the univariate probability for PBIL-iUMDA is straightforward because the model does not depend on how the solutions are ordered. However, under the same assumptions, for PBIL-OS there would be a different value of $p(x_j)$ for each of the $\binom{|S|}{k}$ manners of ordering a vector with k ones. This fact determines that, if the order of the selected solutions is randomly set, in each generation there will be a very high variability in the models learned by PBIL-OS, and in the solutions sampled from these models. Therefore, it is an important question to determine what the expected value of $p(x_i)$ is when all possible orderings of solutions in the selected population are considered. Producing an expression for the univariate probabilities allows us to study the effect of the parameters of PBIL-OS. Furthermore, we can compare PBIL-OS and PBIL-iUMDA in terms of the expected values of the univariate probabilities.

In the following lines, we derive an expression of the univariate probability ($\bar{p}(x_i)$) of variable x_i when all possible orderings of the selected population are considered:

$$\bar{p}(x_j) = \frac{\binom{|S|}{k} b(1-\alpha)^{|S|} + \alpha \sum_{j=1}^{\binom{N}{k}} \sum_{l=1}^{|S|} x_j^i (1-\alpha)^{|S|-l}}{\binom{|S|}{k}} \tag{4.6}$$

$$= \frac{\binom{|S|}{k} b(1-\alpha)^{|}S| + \frac{k\binom{|S|}{k}}{|S|}\alpha \left(\sum_{l=0}^{|S|-1}(1-\alpha)^{|S|-l-1}\right)}{\binom{|S|}{k}} \tag{4.7}$$

$$= b(1-\alpha)^{|S|} + \frac{k}{|S|}\alpha\left(\frac{1-(1-\alpha)^{|S|}}{1-(1-\alpha)}\right) \tag{4.8}$$

$$= b(1-\alpha)^{|S|} + \frac{k}{|S|} - \frac{k}{|S|}(1-\alpha)^{|S|} \tag{4.9}$$

$$= \frac{k}{|S|} + (1-\alpha)^{|S|}(b - \frac{k}{|S|}) \tag{4.10}$$

For the derivation, we have used the fact that the number of non-zero values in each of the $|S|$ positions in the population is $\frac{k\binom{|S|}{k}}{|S|}$, and the exponential sum formula is $\sum_{l=0}^{|S|-1} c^i = \frac{1-c^{|S|}}{1-c}$.

We can see in Equation (4.10) that the expected value of the univariate probability when all possible orderings are considered can be very close to $\frac{k}{|S|}$, particularly for large values of $|S|$. If the initial probability b is $\frac{k}{|S|}$, then the expected value $\bar{p}(x_j)$ is exactly $\frac{k}{|S|}$.

4.3.1 Simulations

In this section we use the expressions derived for the expected univariate probabilities of the models learned by PBIL-iUMDA and PBIL-OS to investigate the influence of different parameters in the probabilities learned by the two variants of PBIL.

Figure 4.1 shows the results of the simulation of the update mechanism of PBIL-OS using $b = 0.5$ and $\alpha = 0.7$. It shows the output $p(x_j)$ values for all possible $\binom{|S|}{k}$ orders and different values of k. The population has $|S| = 10$ individuals and, in the column vector, there are exactly k variables with value 1, i. e., $\sum_{i=1}^{|S|} x_j^i = k$. It can be seen in Figure 4.1 (left) that, although the 120 orderings produce probabilities very different to $\frac{k}{|S|} = 0.3$, there are some common patterns between them. As expected, a larger range of values is obtained for $k = 5$ (Figure 4.1, right).

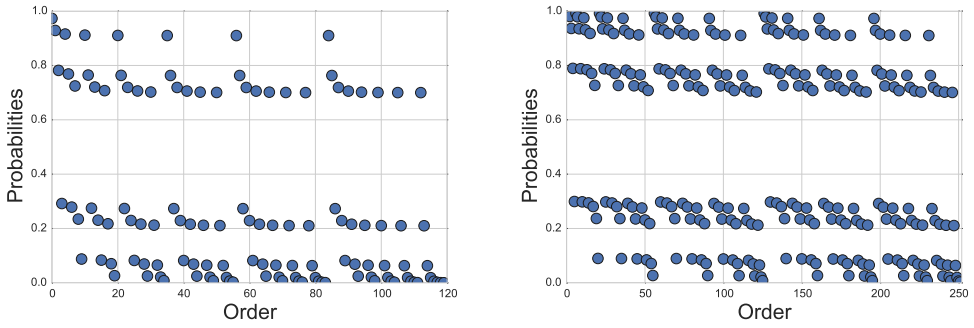


Figure 4.1: $p(x_j)$ values of the update mechanism of PBIL-OS for all possible orders using $k = 3$ (left) and $k = 5$ (right).

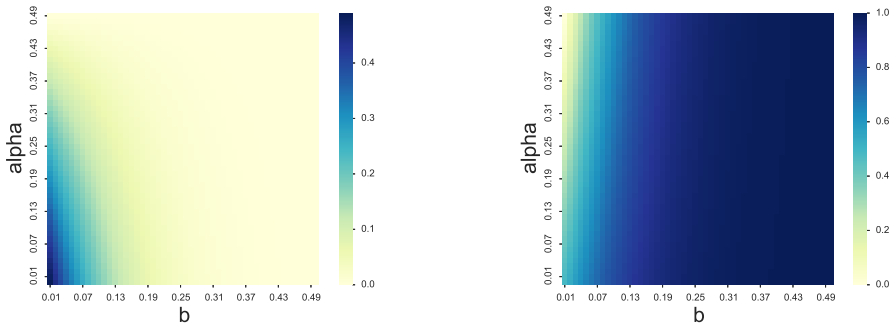


Figure 4.2: $p(x_j)$ values of the update mechanism of PBIL-OS when initial parameters b and α, are changed for two possible orders ($k = 3$). The first k solutions in the order have value $x_j = 1$ (left). The last k solutions in the order have value $x_j = 1$ (right)

The derived equations can be also used to investigate the effect that different parameters of PBIL-OS have in particular orderings of the solutions. For example, in Figure 4.2 we explore the effect of $b \in [0, 0.49)$ and $\alpha \in [0, 0.49)$ in two different orderings of 10 solutions with $k = 3$. The first ordering, for which results are shown in Figure 4.2 (Left) corresponds to the situation in which the first k solutions in the population are those that have $x_j = i$. In Figure 4.2 (Right) the ordering is the reverse. The last k solutions are those that have $x_j = 1$.

It can be seen in Figure 4.2 how the effect of the same combination of parameters is completely different depending on the ordering of the same solutions. For a large number of combinations of b and α the probabilities of the first ordering will converge to 0. For the second ordering the effect is the opposite. These results indicate that the effect of the PBIL-OS parameters can not be considered independently of the criterion chosen to sort the selected solutions. In the next section, we analyze this question from an empirical point of view.

4.4 Experiments: single-objective case

We can assume that solutions in the selected populations of PBIL-OS follow a given order, for example, solutions can be sorted according to the fitness values. Then, the question arises of whether a particular order of the solutions is more beneficial in terms of the optimization results. In the following, the three different PBIL learning mechanism are described:

1. The PBIL-OS-d, in which the solutions in selected population (S) are sorted according to the fitness value from the best solution to the worst (i.e., descending order)

2. The PBIL-OS-a is the reverse order of the selected solutions in S.

3. PBIL-iUMDA, which is the third variant.

We have selected four representative functions which serve to investigate the different facets of difficulty for EAs [Mühlenbein e Paass, 1996]: *One-Max*, $Trap_3$, *Checkerboard*, and *FourPeaks*. These functions were also used in the original paper of PBIL [Baluja, 1994] and have been widely addressed in the literature to evaluate different EDAs [Echegoyen et al., 2007, Mendiburu et al., 2012].

Let $u(\mathbf{x}) = \sum_{i=1}^{n} x_i$ be a function defined in terms of its number of "1s". $f(\mathbf{x})$ is a unitation function if $\forall \mathbf{x},\mathbf{y} \in \{0,1\}^n$ and $u(\mathbf{x}) = u(\mathbf{y}) \rightarrow f(\mathbf{x}) = f(\mathbf{y})$.

The function *One-Max* is defined as:

$$\textit{One-Max}\ (\mathbf{x}) = \sum_{i=1}^{n} x_i = u(\mathbf{x}) \tag{4.11}$$

Unitation functions are also useful for the definition of deceptive functions. In deceptive functions, the difficulty is given by the interactions that arise among subsets of variables. The $Trap_3$ is an additively separable (non-overlapping) function with a unique optimum. It divides the vector $\mathbf{x}$ on p disjoint vectors x_I of 3 variables.

$$Trap_3(\mathbf{x}) = \sum_{I=1}^{n/3} trap3(x_I) \tag{4.12}$$

where

$$trap3(x_I) = \begin{cases} 3, & \text{if } u(x_I) = 3 \\ 2-u & \text{if } u < 3 \end{cases} \tag{4.13}$$

This function has one global optimum and a large number of local optima, particularly $2^{n/3} - 1$.

For *Checkerboard*, the goal is to create a checkerboard pattern of 0's and 1's in an $N \times N$ grid. For each position in an $(N-2)(N-2)$ grid centered on an $N \times N$ grid, +1 is added for

each position whose four primary neighbors (above, below, left, right) have the opposite value to that position.

The function *FourPeaks* is defined as:

$$FP(x,t) = max\{head(1,x), tail(0,x)\} + R(x,t) \tag{4.14}$$

where $head(1,x)$ is the number of leading 1's in x, $tail(0,x)$ is the number of trailing 0's in x and the reward value $R(x,t)$ is 100 if $head(1,x) > t$ and $tail(0,x) > t$, otherwise $R(x,t) = 0$. The goal is to maximize the function. The global optimum is difficult to reach because the two local optima (0, 0, ..., 0) and (1, 1, ..., 1) are easier to reach, i.e., the "hill-climbing" search gets trapped quickly in these local optima. By increasing t, the bias of attraction surrounding the inferior local optima increases in size exponentially, while the bias around the global optima decreases.

4.4.1 Parameter Settings

We have investigated the behavior of PBIL variants using three values for the learning rate $\alpha \in \{0.01, 0.05, 0.1\}$. These α values have been commonly used in the analyses of PBIL variants [Mendiburu et al., 2006]. Also, we have used two values for the population size, $N \in \{50, 100\}$. The truncation selection is set to $|S| = N/2$. The stop criterion is 500 generations.

We have used different problem sizes to generate a large set test functions. The parameters settings are: (i) *One-Max*: $n = (100, 250, 500)$; (ii) *Checkerboard*: $n = (36, 64, 100)$; (iii) *FourPeaks*: $t = (\%20, \%30)$ of $n = 100$; and (iv) $Trap_3$: $n = (102, 240, 360)$.

4.4.2 Comparison results

We have executed 30 runs for each combination PBIL variant × test function × parameter setting. The best fitness solution (after the 500 generations) is used as the criterion to evaluate the algorithms.

In addition, for each comparison result, we have applied the *Kruskal-Wallis* test [Derrac et al., 2011] to check whether the final best solutions obtained by the PBIL variants have a significant differences at 0.05 of significance level. For each comparison result, the algorithms are ranked from "1" (which represents the best algorithm result) to "3" (which represents the worst). If two or more algorithms have the same rank, it means that they are not significantly different.

Tables 4.1,4.2,4.3 and 4.4 present the *Kruskal-Wallis* ranking for each combination of parameters setting × test function. Figure 4.3 and 4.4 illustrate the behavior of the algorithms throughout the generations. We make the following remarks:

The results from Tables 4.1-4.4 reveal that, according to the *Kruskal-Wallis* test and under the same conditions, at least two PBIL variants achieved a significant difference in their final results in 63 of the 66 cases. These results show that the PBIL search ability is affected depending on how the learning mechanism is implemented. Also, the three variants have different behaviors depending on the value of the learning parameter α. For the lowest values ($\alpha = 0.01$ and $\alpha = 0.05$), PBIL-OS-d achieves significantly better results than PBIL-OS-a in 24 of the 44 comparison results, and significantly better than PBIL-iUMDA in 16 of the 44 cases. For the highest value ($\alpha = 0.1$), PBIL-iUMDA achieves significantly better results than PBIL-OS-d in 9 of the 22 comparison results and better than PBIL-OS-a in all of the 22 cases. Also, the differences are more significant when n is larger.

Table 4.1: Statistical ranking of the PBIL variants according to the *Kruskal-Wallis* test for the *One-Max* optimization function

n	N	α	PBIL-OS-d	PBIL-OS-a	PBIL-iUMDA
100	50	0.01	1.5	1.5	3
		0.5	2	2.5	1.5
		0.1	2	3	1
	100	0.01	1.5	1.5	3
		0.5	2	2	2
		0.1	1.5	3	1.5
250	50	0.01	1.5	1.5	3
		0.5	2	3	1
		0.1	2	3	1
	100	0.01	1.5	1.5	3
		0.5	1.5	3	1.5
		0.1	2	3	1
300	50	0.01	1.5	1.5	3
		0.5	2	3	1
		0.1	2	3	1
	100	0.01	1.5	1.5	3
		0.5	2	3	1
		0.1	2	3	1

Table 4.2: Statistical ranking of the PBIL variants according to the *Kruskal-Wallis* test for the *Checkerboard* optimization function.

n	N	α	PBIL-OS-d	PBIL-OS-a	PBIL-iUMDA
36	50	0.01	1.5	3	1.5
		0.05	1.5	3	1.5
		0.1	1.5	3	1.5
	100	0.01	2	2	2
		0.05	2	2	2
		0.1	1.5	3	1.5
64	50	0.01	1.5	1.5	3
		0.05	1.5	3	1.5
		0.1	1.5	3	1.5
	100	0.01	1.5	1.5	3
		0.05	1.5	3	1.5
		0.1	1.5	3	1.5
100	50	0.01	1.5	1.5	3
		0.05	1.5	3	1.5
		0.1	1.5	3	1.5
	100	0.01	1	2	3
		0.05	1.5	3	1.5
		0.1	1.5	3	1.5

Table 4.3: Statistical ranking of the PBIL variants according to the *Kruskal-Wallis* test for the *FourPeaks* optimization function.

t	N	α	PBIL-OS-d	PBIL-OS-a	PBIL-iUMDA
20	50	0.01	1	2	3
		0.05	1.5	3	1.5
		0.1	2	3	1
	100	0.01	1	2	3
		0.05	1.5	3	1.5
		0.1	2	3	1
30	50	0.01	1	2.5	2.5
		0.05	1	3	2
		0.1	2	3	1
	100	0.01	1	2	3
		0.05	1	3	2
		0.1	2	3	1

Figure 4.3 and 4.4 clearly show that the three variants have different search behaviors throughout the generations. PBIL-OS-d and PBIL-OS-a have different behaviors in terms of

Table 4.4: Statistical ranking of the PBIL variants according to the *Kruskal-Wallis* test for the $Trap_5$ optimization function.

n	N	α	PBIL-OS-d	PBIL-OS-a	PBIL-iUMDA
102	50	0.01	1.5	1.5	3
		0.5	2.5	2.5	1
		0.1	2	3	1
	100	0.01	1.5	1.5	3
		0.5	2.5	2.5	1
		0.1	2.5	2.5	1
140	50	0.01	1.5	1.5	3
		0.5	2	3	1
		0.1	2	3	1
	100	0.01	1.5	1.5	3
		0.5	1	2	3
		0.1	2	3	1
360	50	0.01	1.5	1.5	3
		0.5	1	2.5	2.5
		0.1	2	3	1
	100	0.01	1.5	1.5	3
		0.5	1	2	3
		0.1	2	3	1

convergence and diversity. Recall that the only difference between them is the order in which their selected populations S are sorted. In the first generations, PBIL-OS-a converges faster than the other variants but, after a number of generations, it easily falls in local optima. Thus, we can conclude that, in PBIL-OS, when the probabilistic vector is updated using the solutions in S sorted from the worst fitness solution to the best fitness solution it has a detrimental effect leading to premature convergence of the model. Moreover, the lowest α value has the most detrimental effect for the PBIL-iUMDA, leading to results far from the optimum after the 500 generations.

A low α value means that the algorithm needs more generations to converge to the optimum. However, a high α value leads to a premature convergence of the probabilistic vector, and consequently to a lack of diversity in the population.

4.5 Experiments: multiobjective case

As an approach to evaluate the PBIL variants in the multiobjective case, we have incorporated them into our MOEA/D framework. In this case, the algorithm maintains N probabilistic models, in which each subproblem i learns and samples a probabilistic model p^i, using the $B(i)$ as the selected population.

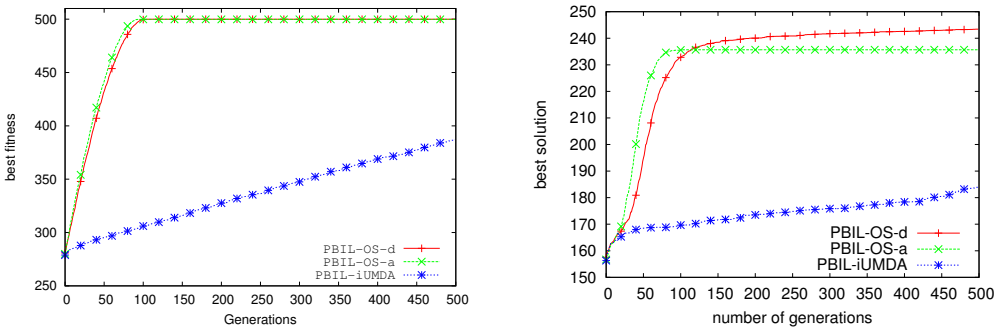
Furthermore, we present the literature review of EDAs within MOEA/D in Section 6.3 where we provide a more intensive discussion about the use of probabilistic models within MOEA/D.

The PBIL learning mechanisms incorporated into our MOEA/D framework are described in the following.

1. The conventional strategy used in MOEA/D that sorts $B(i)$ from the closest neighbor to the farthest. We call this variant as MOEA/D-PBIL-OS-c.

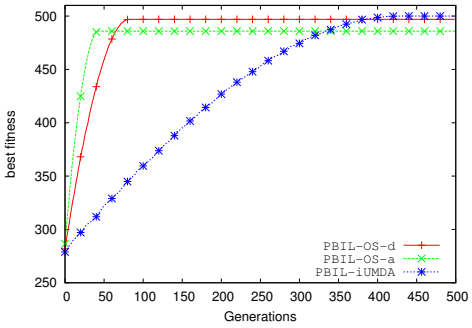
2. The solutions in $S = B(i)$ are sorted according to the scalar aggregation function values, in such a plan that $S = B(i) = \{g(x^{best}|\lambda^i), \ldots, g(x^{worst}|\lambda^i)\}$. We call this variant as descendant order (MOEA/D-PBIL-OS-d).

3. The third variant is the reverse order of PBIL-OS-d, called MOEA/D-PBIL-OS-a.

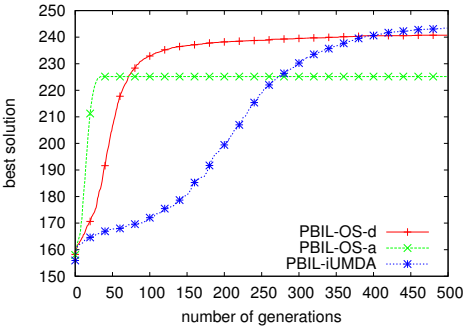


(a) *One-Max* $\alpha = 0.01$

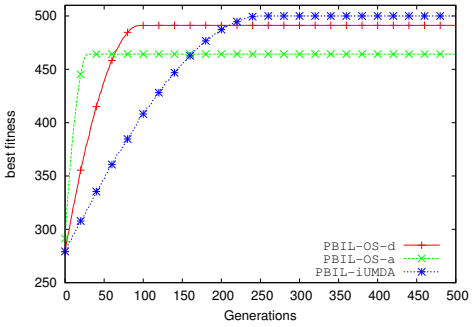
(b) *Checkerboard* $\alpha = 0.01$



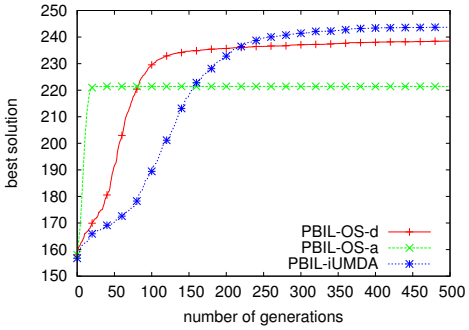
(c) *One-Max* $\alpha = 0.05$



(d) *Checkerboard* $\alpha = 0.05$



(e) *One-Max* $\alpha = 0.1$



(f) *Checkerboard* $\alpha = 0.1$

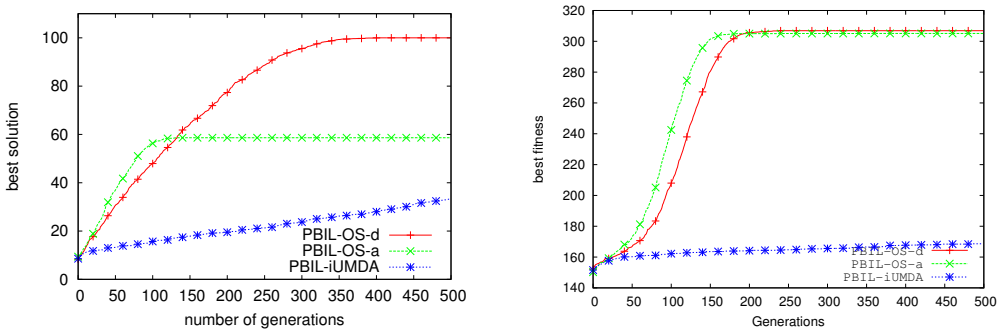
Figure 4.3: Behavior of PBIL-OS-d, PBIL-OS-a and PBIL-iUMDA throughout the generations for the *One-Max* (a) (c) and (e) with $n = 500$, $N = 100$; and *Checkerboard* (b) (d) (f) with $n = 100$, $N = 100$.

4. The last strategy is the MOEA/D-PBIL-iUMDA.

We have used the instances from the mUBQP repository [Liefooghe et al., 2014]. The instances vary according to a fixed number of variables $n = 1000$, two objectives ($m = 2$), and the correlation strength are $\rho = \{-0.5, 0.0, 0.5\}$ (from the weakest to the strongest correlation, respectively). We used the Hypervolume indicator and the cardinality to evaluate the outcomes. The Kruskall-Wallis is applied to evaluate if there exist significant difference between them.

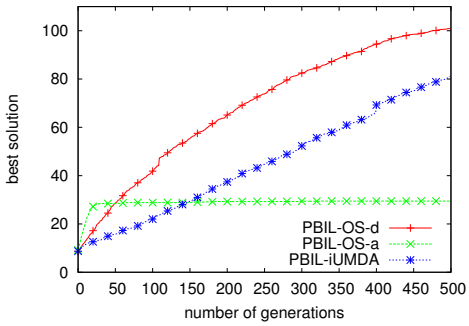
4.5.1 Parameter Settings

1. *Number of subproblems:* As in the previous chapter, $N = 201$.
2. *Neighborhood size*: As the number of selected solutions is crucial for EDAs, we test two different values $T = \{20, 40\}$.

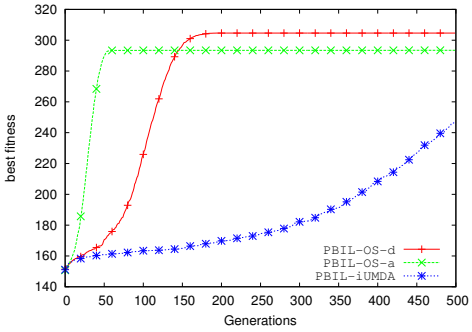


(a) *FourPeaks* $\alpha = 0.01$

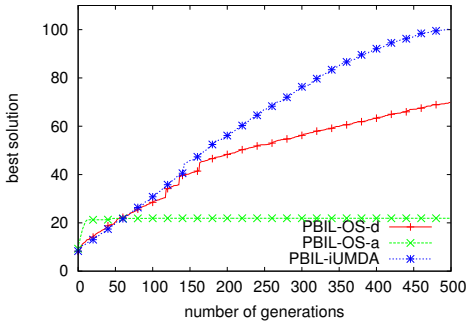
(b) $Trap_3$ $\alpha = 0.01$



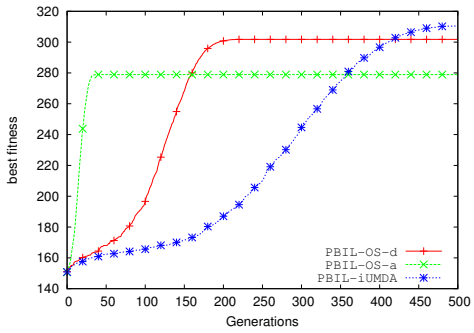
(c) *FourPeaks* $\alpha = 0.05$



(d) $Trap_3$ $\alpha = 0.05$



(e) *FourPeaks* $\alpha = 0.1$



(f) $Trap_3$ $\alpha = 0.1$

Figure 4.4: Behavior of PBIL-OS-d, PBIL-OS-a and PBIL-iUMDA throughout the generations for the *FourPeaks* (a) (c) and (e) with $n = 100$, $N = 100$, $t = 30$; and $Trap_3$ (b) (d) (f) with $n = 360$, $N = 100$

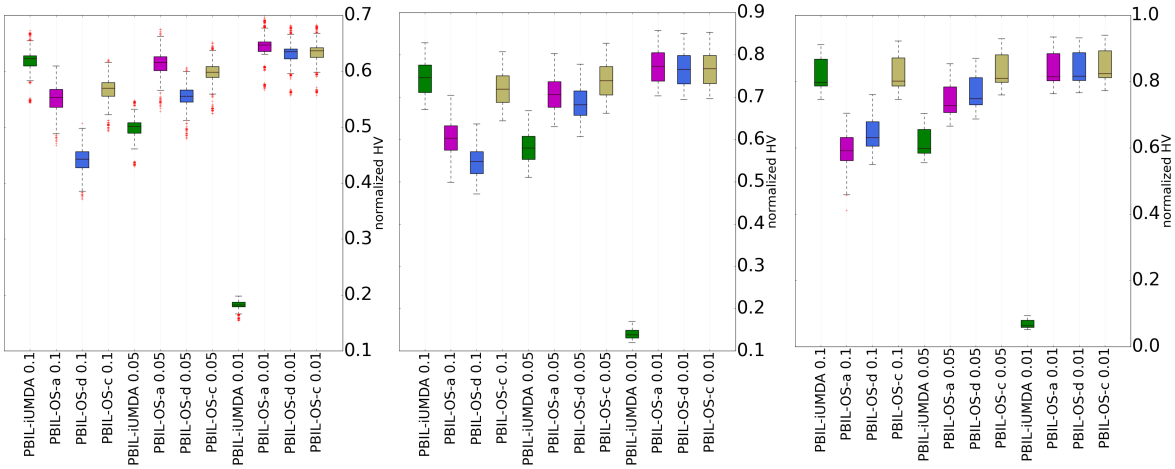
3. *Maximal number of replacement by a new solution:* As in the previous chapter, $n_r = 2$.

4. *PBIL learning rate:* We used the same α that was used in the single-objective case $\alpha = \{0.01, 0.05, 0.1\}$.

Each test instance is independently run 30 times. The stopping condition for the algorithms was 500 generations.

4.5.2 Comparison results

Figure 4.5 presents the normalized *HV* measure obtained. Table 4.5 presents a *Kruskall-Wallis* ranking test (at 5% significance level) of the algorithms for each combination of parameters $\times$ test instance. Table 4.6 presents the mean and the standard deviation of the size of the approximated *PFs* size.



(a) $\rho = -0.5$ (b) $\rho = 0.0$ (c) $\rho = 0.5$

Figure 4.5: Box plot of all normalized *HV* values obtained for the PBIL variants using $T_m = 40$ for the mUBQP instances with $\rho = \{-0.5, 0.0, 0.5\}$. Each column of a plot is a combination of PBIL variant $\times$ $\alpha = \{0.01, 0.05, 0.1\}$

Table 4.5: *Kruskal-Wallis* ranking according to the *HV* measure

ρ	α	T_m	PBIL-OS-c	PBIL-OS-d	PBIL-OS-a	PBIL-iUMDA
-0.5			2.5	1.5	2	4
0.0	0.01		2	2	2	4
0.5			2	2	2	4
-0.5			1	2.5	2.5	4
0.0	0.05	20	1	2.5	2.5	4
0.5			1	2.5	2.5	4
-0.5			1	4	3	2
0.0	0.1		1	3.5	3.5	2
0.5			1	3	4	2
-0.5			2.5	2.5	1	4
0.0	0.01		2	2	2	4
0.5			1	2.5	2.5	4
-0.5			2	3	1	4
0.0	0.05	40	1	3	2	4
0.5			1	2	3	4
-0.5			2	4	3	1
0.0	0.1		1.5	4	3	1.5
0.5			1	3	4	2

In terms of the *HV*, Figure 4.5 and Table 4.5 show that: In all the cases, there is, at least, one result with significant difference. The differences among PBIL-OS(-c,-d,-a) increase when $T_m = 40$ and when the α value is greater. Also, PBIL-OS-c (conventional sorting according to the neighborhood Euclidean distance) achieved the best rank in 10 of the 18 cases, which means that learning from the closest to the farthest neighbor is efficient for the search process throughout the generations.

In terms of the final approximated *Pareto* front sizes, the three PBIL-OS variants produce more populated approximated *PFs* than PBIL-iUMDA. As in the single-objective case, the results seem to indicate that (in the same conditions) PBIL-iUMDA is slower than PBIL-OS variants

Table 4.6: Mean (standard deviation) of the Pareto fronts final size, $T_m = 40$

ρ	α	PBIL-OS-c	PBIL-OS-d	PBIL-OS-a	PBIL-iUMDA
-0.5	0.01	366.67 (36.78)	433.25 (52.55)	436.38 (52.62)	43.19 (5.99)
0.0		288.50 (20.71)	335.14 (27.76)	334.14 (24.66)	23.70 (4.45)
0.5		195.26 (15.30)	200.79 (15.27)	202.61 (14.15)	11.63 (3.21)
-0.5	0.05	386.92 (39.0)	601.61 (97.75)	537.13 (56.94)	111.50 (9.22)
0.0		296.72 (23.34)	456.33 (45.66)	349.87 (23.67)	69.89 (7.04)
0.5		197.07 (16.41)	195.35 (13.22)	231.84 (18.77)	39.89 (5.3)
-0.5	0.1	356.81 (39.65)	851.56 (147.16)	464.11 (61.05)	230.18 (14.90)
0.0		283.09 (24.55)	596.41 (91.63)	385.10 (48.25)	168.59 (12.06)
0.5		183.14 (15.6)	187.27 (14.25)	261.65 (42.75)	112.09 (9.98)

to explore the search space and only achieves competitive results using a higher learning rate ($\alpha = 0.1$)

As in the analytical study, our results show that the different implementations of PBIL produce a high variability in the vectors of probability which have an impact on the behavior of the algorithms in terms of convergence and diversity during the search throughout the generations.

4.5.3 Comparison to MOEA/D-BACO

In this section, we have intended to compare the MOEA/B-BACO (proposed in the previous chapter) and MOEA/D-PBIL-OS for solving the mUBQP. Here, we have considered the MOEA/D-PBIL-c. Theoretically, both algorithms are sensitive to the quality function but in different actions. MOEA/D-BACO gives proportionally more pheromone amount (update) for the highest quality solutions, while the selected solutions in MOEA/D-PBIL is sorted according to their quality assessment and this explicit sorting affects the search behavior.

The remaining parameters setting are as follows: neighborhood size $T = 20$, BACO learning rate $\alpha = 0.1$, PBIL learning rate $\alpha = 0.01$, maximum replacements $n_r = 2$, and the stopping condition $time = n \times m \times 300$ milliseconds, and the experiment was performed on a PC with Intel Xeon E5-620 2.4 GHz processor and 12 GB memory. We have not used the UBQP local search in this analysis.

The results in Table 4.7 show that MOEA/D-PBIL-OS-c significantly outperforms MOEA/D-BACO after $n \times m \times 300$ milliseconds in 8 of the 10 cases. The MOEA/D-PBIL-OS-c have less computational cost (complexity) than -BACO. As well as, the algorithms are too sensitive to the parameter α, which can control the search process.

In this case, we have not compared these results to the *best-known* results [Liefooghe et al., 2015] because we have used a limited computational cost as the stopping condition instead of a maximum number of generations. Therefore, a deeply comparison of MOEA/D-PBIL-c and the other variants against the *best-known* approximated *PFs* can be investigated in the future.

Table 4.7: Average HV results after $time = n \times m \times 300$ ms

Instance		$m = 2$	
n	ρ	MOEA/D-PBL-c	MOEA/D-BACO
	-0.5	**0.6977**	**0.6707**
	-0.2	**0.7434**	0.6736
	0.0	**0.7682**	0.6513
	0.2	**0.8257**	0.6507
	0.5	**0.8322**	0.5723
1000		$m = 3$	
	-0.5	**0.4824**	**0.4481**
	-0.2	**0.4709**	0.3997
	0.0	**0.5091**	0.3835
	0.2	**0.5026**	0.3412
	0.5	**0.5574**	0.2980

4.6 Conclusions and future work

Understanding the behavior of an EDA plays a vital role for its efficient application to different problems, and this understanding enables us to produce better results. Due to the simplicity of the model used by PBIL, and its low computational overhead, the algorithm has been applied to a variety of problems. In this chapter, we have identified the two main learning strategies that define PBIL applications reported in the literature. To the knowledge of the authors, no previous research has addressed the question of how differences in the implementation of the PBIL learning mechanism determine completely different behaviors of the algorithm.

Being one of the most applied EDAs, this is a relevant aspect because, as we report in this chapter, the different implementations of the PBIL's learning mechanism have an important effect on their search behavior. We have studied it analytically, by deriving the expected value of the probability vector for PBIL-OS and PBIL-iUMDA. The analysis shows that, while the expected univariate probabilities of PBIL-iUMDA do not depend on the order of the solutions, the univariate probability distributions of PBIL-OS are highly influenced by the form in which the same set of solutions are sorted.

The empirical results show that the diversity produced by the variability inherent to the PBIL variants has a direct effect on the search ability of the algorithm. In general, in PBIL-OS, the last solution has a stronger impact on the final configuration of the probability vector. Thus, ordering solutions from the best to the worst, contributes to the exploration. Furthermore, we show that the best choice of α for PBIL-OS usually produces poor results for PBIL-iUMDA.

These finds is useful in some aspects. Firstly, when applying PBIL, practitioners will be aware that two different variants of the algorithm exist, and could make an informed choice of the appropriate variant. Besides, when applying the PBIL-OS variant, users will know that the way selected solutions are sort to update the PBIL model has a strong impact in the behavior of the algorithm, which can be used to bias learning. Newly, introduced algorithms should be described to a great level of detail to avoid multiple interpretations and allow reproducibility.

Chapter 5

On the design of hard mUBQP instances

5.1 Introduction

One of the avenues to understand the difficulties that MOPs represent for MOEAs is the design of benchmark functions able to represent different facets of complexity of MOPs. A number of benchmarks have been proposed in the literature [Deb et al., 2005, Huband et al., 2006] and they continue to be extensively used for evaluating MOEAs. Most of test functions are mainly focused on continuous problems. However, research on combinatorial MOPs can provide insights about how the structural characteristics of the objective functions influence the shape and other attributes of the Pareto fronts [Aguirre e Tanaka, 2007, Verel et al., 2011, Verel et al., 2013].

The method to generate mUBQP instances, proposed in [Liefooghe et al., 2014], ensures that instances will satisfy a number of characteristics related to the sparsity of the single-objective problems and the degree of correlation between the objectives. Both characteristics have an important impact on the behavior of MOEAs.

On the other hand, average-sense misleading (deceptive) functions [Deb e Goldberg, 1993] have played a fundamental role in the analysis of GAs. In deceptive functions, there are at least two optima, a true optimum and a deceptive optimum. Commonly applied search operators tend to favor the deceptive optimum making very difficult for the algorithm to find the true optimum. Pelikan et al. [Pelikan et al., 2005] used the simple bi-objective deceptive function for analyzing the behavior of hBOA and other MOEDAs. The bi-objective trap problem comprises unitations functions $trap_5$ (Equation 6.1) and $invtrap_5$ (Equation 6.2). These function are conflicting, and since the global optimum is reached in a single point, for a problem of $n = 5l$ variables, the number of solutions in the Pareto set is 2^l.

In this chapter, we follow a different path to the construction of mUBQP instances. Our proposal is based on planting deceptive modules in the matrices describing the structure of the objective functions. We have used the $trap_5$ and the $invtrap_5$ functions as inspiration to propose a way to construct deceptive pairs of UBQP submatrices.

The chapter is organized as follows: Section 5.2 discusses the related work. Our approach to construct instances of the mUBQP is explained in Section 5.3. Section 5.4 summarizes the process used to generate functions of varying difficulty. Section 5.5 presents the numerical results. The main contributions are summarized in Section 5.6.

5.2 Related work

A number of papers have proposed benchmarks for evaluating different facets of the difficulty of MOPs [Deb et al., 2005, Huband et al., 2006, Li e Zhang, 2009]. These benchmarks mainly comprise continuous functions and emphasis is put on the shape of the Pareto front and not on the relationships between the decision variables.

Knowles and Corne [Knowles e Corne, 2003] introduced an instance generator for the multi-objective quadratic assignment problem (mQAP) and investigated the effect of correlations between the objectives. To enforce the correlations between the mQAP objectives, a correlation parameter c is used. Based on this fixed parameter, the instances are generated.

Aguirre et al. [Aguirre e Tanaka, 2007] present an exhaustive investigation of how the parameters of the MNK-landscape influence several characteristics of the fitness landscape, including the size of the Pareto front and the number of fronts.

Verel et al. [Verel et al., 2011, Verel et al., 2013] extended the NK-model to design multi-objective functions with correlations. In the ρMNK-landscapes, the epistatic structure is the same for all the objectives, and the local fitness functions are not independent. In [Verel et al., 2011], a complete enumeration of the small ρMNK-landscape instances was used to investigate the effect of the correlations in the characteristics of the Pareto front. In a recent work, the ρMNK-landscape model has been used to investigate the influence of the parameters describing the landscapes in the behavior of different MOEAs [Marquet et al., 2014], showing the general usefulness of this type of models. The approach followed to create instances is different to the one presented here.

Much work in evolutionary computation has been devote to study deception for combinatorial problems. The proposal presented in this chapter is inspired by previous analysis of multi-objective deceptive functions [Pelikan et al., 2005, Martins et al., 2011].

5.3 Planting patches of difficult subfunctions in mUBQP matrices

Regarding the mUBQP formulation described in Section 3.2, Liefooghe et al. [Liefooghe et al., 2014] have proposed an approach to construct mUBQP instances according to some properties. Each objective function $k \in \{1, .., m\}$ is defined by means of a matrix $Q^k = (q^k_{ij})$ of size $n \times n$ with constant positive, negative or zero values. The *search space* Ω is defined on binary strings of size n. In the single-objective case, the density d gives the expected proportion of non-zero number in the matrix. In order do define m matrices of a given density d, they set $q^k_{ij} = 0$ for all objectives at the same time. The correlation between the data contained in the m matrices is set using the same correlation between all pair of objectives, given by a correlation coefficient $p > \frac{-1}{m-1}$. The positive (respectively negative) coefficient decreases (respectively increases) the degree of conflict between all pair of objectives. These properties have a direct effect on the number of solutions in the *PFs*.

In the following, we exemplify a bi-objective mUBQP instance (matrices Q^1 and Q^2) with a density of non-zero $d = 0.9$, and a negative objective correlation $\rho = -0.9$. The problem size is $n = 3$, consequently there is 3×3 coefficients q_{ij} for each objective function.

$$Q^1 = \begin{pmatrix} 67 & -8 & 0 \\ -92 & -53 & -32 \\ 0 & -96 & 0 \end{pmatrix} \qquad Q^2 = \begin{pmatrix} 30 & 20 & 0 \\ -73 & -34 & 77 \\ 0 & 7 & 0 \end{pmatrix} \tag{5.1}$$

We have followed a different strategy to create new mUBQP instances. We propose a bottom-up algorithm in which the mUBQP matrices are filled by the addition of small deceptive sub-matrices.

5.3.1 Parametrization of the UBQP problem

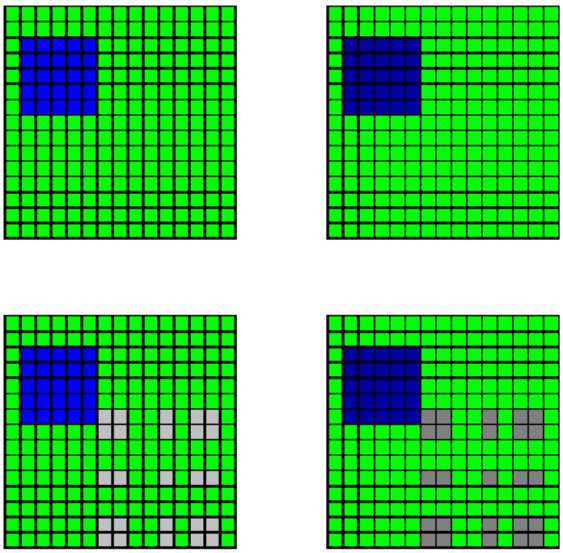


Figure 5.1: Two steps in the construction of an instance of a bi-objective UBQP problem ($n = 15$). Top row: two compact submatrices ($n_p = 5$) containing the relationships between variables $\{X_2, X_3, X_4, X_5, X_6\}$ are planted. Bottom row: two submatrices ($n_p = 5$) are planted containing the relationships between variables $\{X_7, X_8, X_{11}, X_{13}, X_{14}\}$.

The algorithm starts from m sets of zero matrices and, in each step, a single sub-matrix is added to each of the m matrices. Each sub-matrix is "planted" in the same position of the m mUBQP matrices. The process is illustrated in Figure 1, where we show an example of a bi-objective problem of size $n = 15$. In the figure, green cells corresponding to matrix entries equal to zero. In the first step (top matrices), there is one sub-matrix ($n_p = 5$) added to each matrix. Although each pair of sub-matrices is added in the same location for each matrix, the sub-matrices values are different between them. In the second step (bottom matrices), another pair of sub-matrices is added. Notice that sub-matrices do not need to contain sequential variables. For instance, the second set of sub-matrices shown in Figure 6.1 contains variables $\{X_7, X_8, X_{11}, X_{13}, X_{14}\}$.

There are a number of key issues relevant to our approach: 1) How are the submatrices created? 2) How to select the position to insert the submatrices? 3) How many submatrices to add? In the following, we discuss these issues.

To create a sub-matrix, we parameterize $n_p < n$ variables of the problem (single-objective) in terms of interactions between their q_{ij}. For instance, we set $n_p = 5$. We constrain the possible values each value can take and set $q_{ij} \in \{-1, 1\}$ and $q_{ij} = 0$ if $i = j$. For $n_p = 5$, there are $\frac{n_p(n_p-1)}{2} = 10$ pairwise interactions contained in this submatrix. Considering the two values each variable can take, the total number of possible UBQP functions that can be generated is $|\mathcal{F}| = 2^{10} = 1024$, where $\mathcal{F}$ is the set of functions, and each function corresponds to a different

submatrix. Equation 5.2 shows the expression for the general parameterization of such UBQP problem.

$$Q = \begin{pmatrix} 0 & q_{12} & q_{13} & q_{14} & q_{15} \\ q_{12} & 0 & q_{23} & q_{24} & q_{25} \\ q_{13} & q_{23} & 0 & q_{34} & q_{35} \\ q_{14} & q_{24} & q_{34} & 0 & q_{45} \\ q_{15} & q_{25} & q_{35} & q_{45} & 0 \end{pmatrix} \tag{5.2}$$

It is important to emphasize that, we refer here to the space of UBQP matrices that satisfy the constraints of symmetry ($q_{ij} = q_{ji}$) for a very reduced number of values of interactions ($q_{ij} \in \{-1, 1\}$). Each of the 1024 defines a different problem of size $n = 5$ and therefore, for each of such problems we could find an optimal binary solution $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$ that maximizes Equation (3.1) for this particular choice of matrix Q.

5.3.2 Computing the degree of deception

We start from all possible functions obtained by the parameterization proposed in Section 5.3.1. From the space of $|\mathcal{F}| = 2^{10} = 1024$ functions, we would like to identify which are deceptive.

First, we need a more formal definition of deception [Whitley, 2015]. Let h denote an $(n - j)$-dimensional hyperplane where j variables have preassigned bit values and $\alpha(h)$ be a mask with 1 bits marking the locations where the j variables appear in the problem location and 0 elsewhere. Let $MAX(\mathbf{x}, \alpha(h))$ return the hyperplane with the best mean over all 2^j order j hyperplanes that can be defined using the $\alpha(h)$ mask.

A function is *order-j deceptive* [Whitley, 2015] if the j bit values returned by $MAX(\mathbf{x}, \alpha(h))$ for all hyperplanes of *order-j* are not the same as the bit values found in a string which is a global optimum.

We use, a more relaxed definition of an order-j deceptive function, in which a function is *midly order*-1 *deceptive* if the j bit values returned by $MAX(\mathbf{x}, \alpha(h))$ for at least one hyperplane of order j is not the same as the bit values found in a string which is a global optimum.

In the particular case of a mildly *order-1* deceptive function, the definition requires that the function will be deceptive if for at least one of the n order-1 hyperplanes the value of $MAX(\mathbf{x}, \alpha(h))$ will be different to the optimum. For small values of n it is very easy to check if a function is mildly deceptive. We can evaluate the entire space of solutions, compute the order-1 hyperplanes, and check whether there is any hyperplane $MAX(\mathbf{x}, \alpha(h))$ that is not in any global optimum. However, for the construction of the mUBQP instances we are not just interested in determining if a function is deceptive but also in measuring the degree of deception. We propose a form to do this based on the computation of univariate probabilities.

One easy manner to test if a function is deceptive from a sufficiently large population of solutions distributed according to the fitness is by computing the univariate frequencies of the variables. If the most frequent configurations for the single variables are not consistent with the configurations of these variables in the global optimum, then the function is midly order-1 deceptive.

For a given matrix Q, we compute the fitness for all the $2^5 = 32$ configurations of $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$. Using the UBQP fitness values as an energy function, we associate to each

possible solution a probability value $p(\mathbf{x})$ according to the Boltzmann probability distribution. The Boltzmann probability distribution $p_B(\mathbf{x})$ is defined as

$$p_B(\mathbf{x}) = \frac{e^{\frac{g(\mathbf{x})}{T}}}{\sum_{\mathbf{x}'} e^{\frac{g(\mathbf{x}')}{T}}}, \tag{5.3}$$

where $g(\mathbf{x})$ is a given objective function and T is the system temperature that can be used as a parameter to simulate the effect of the strength of selection in the frequency of the solutions. In this work, we use $T = 1$. The Boltzmann or Gibbs distribution has the convenient property of assigning an exponentially higher probability to solutions based on their fitness. It has been used before to study the influence of variables interactions in the behavior of EAs in single-objective problems [Mühlenbein et al., 1999, Santana et al., 2001a] and MOPs [Santana et al., 2015]. By choosing $T = 1$ we set the probabilities to depend only on the fitness function.

Notice that, for small UBQP instances as the ones we use for planting solutions, it is easy to compute the Boltzmann distribution since the size of the search space is small (32). Using the Boltzmann distribution, we can also compute the univariate probabilities for each variable (e.g $p_1(x_1 = 1) = \sum_{x'_1=1} p_B(\mathbf{x}')$). Then, it is possible to compute the probability of the optimum given by the univariate factorization $p_u(\mathbf{x}) = \prod_i p_B(x_i)$.

Algorithm 5.1: Degree of deception for function $f(\mathbf{x})$

1. Compute the Boltzmann distribution $p_B(\mathbf{x})$ of function $f(\mathbf{x})$ on the 2^5 solutions.
2. Compute the univariate probabilities from p_B.
3. For each solution, compute its univariate factorization $p_u(\mathbf{x}) = \prod_i^5 p_B(x_i)$
4. Rank the solutions from the highest to lowest $p_u(\mathbf{x})$
5. The average rank of the optimal solutions is the degree of deception

We use the univariate probability to rank all solutions from the one with the highest probability to the one with the smallest probability. If the univariate factorization gives to the optimal solution the highest probability, then the degree of deception is minimum (1). However, as the rank given by univariate factorization increases it means that the function is more deceptive. Since for $n = 5$ there are $2^5 = 32$ possible solutions, the highest value of deception is 32. Algorithm 5.1 summarizes the steps to compute the degree of deception for a function determined by a matrix Q. We computed the degree of deception for each of the 1024 functions in $\mathcal{F}$. This information is shown in Figure 5.2 where functions are sorted in lexicographical order.

5.4 Deceptive mUBQP instances

To analyze the multi-objective case, we then consider all possible bi-objective functions such that $g_a, g_b \in \mathcal{F}$ and $g_a \neq g_b$. We call this set O, where $|O| = \frac{1024*1023}{2} = 523776$. The idea is to identify which are the functions in O that satisfy conditions similar to the multi-objective trap functions, i.e., they are deceptive for the two single objective problems involved, and the Pareto set is as large as possible.

We computed the Pareto sets of all the bi-objective functions in O. We found that the maximum size of the Pareto fronts for functions in O is 8, and the maximum number of Pareto optimal solutions is 32. Table 5.1 shows the number of generated functions for each size of the Pareto front.

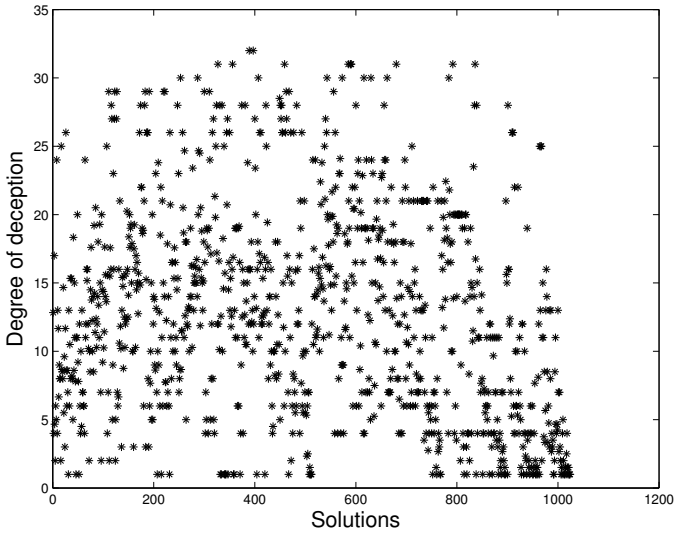


Figure 5.2: Degree of deception for all possible UBQP functions, $n = 5$, $q_{ij} = \{-1, 1\}$.

Table 5.1: Number of bi-objective functions for different sizes of the Pareto fronts

Size PF	1	2	3	4
Numb. functions	94295	190059	150841	66745
Size PF	5	6	7	8
Numb functions	17411	3990	375	60

For a given bi-objective function (g_a, g_b), we compute first the Pareto set. Then, for each Pareto-optimal solution, we compute the degree of deception in each objective. Finally, we define the degree of deception of the bi-objective function (g_a, g_b), as the mean, computed among the objectives, of the average rank of all the solutions in the Pareto-set. This is a straightforward extension of the degree of deception defined for single-objective functions. Other ways to define the degree of deception for MOPs are possible (e.g. considering the minimum degree of deception among the Pareto-optimal solutions, quartile, mode, and other criteria). Figure 5.3 shows the histogram of the degree of deception for the bi-objective functions in O.

We assume that, if the Pareto-optimal solutions are deceptive, then MOEAs will spend more computational effort to find these Pareto-optimal solutions or will miss them.

In the previous section, we have explained how to create bi-objective deceptive functions and compute the degree of deception. We use O as a source of bi-objective functions (submatrices) with a different number of solutions in the Pareto set and different degrees of deception. We will use these submatrices to apply the planting procedure explained in Section 5.3 and create larger deceptive mUBQP instances.

We define a deceptive mUBQP problem using the parameters $\{n, M, d, l\}$ where n is the number of variables, M is the number of objectives, d is the density of zeros in all the M matrices that define the problem, and l is the amount of injected deception. We focus in the bi-objective case, i.e. $M = 2$.

Parameters $\{n, M, d\}$ are used in [Liefooghe et al., 2014], where a generator of mUBQP instances is proposed. We define $l = \frac{k}{n'}$, where k is the number of deceptive UBQP functions ($n = 5$) *injected* to a *traditional* mUBQP instance generated with parameters $\{n', M, d, c\}$, where

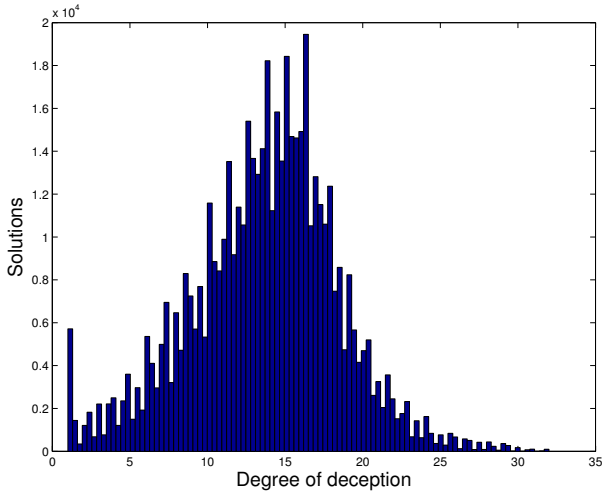


Figure 5.3: Histogram of the degree of deception for functions in O.

$n' >> 5$. The steps to generate the mUBQP instances are shown in Algorithm 5.2. The algorithm receives as input a set of seeding functions $O' \subset O$ explained above.

Algorithm 5.2: Generation of large mUBQP instances

```
1 Initialize matrices Q1 and Q2 to zero.
2 for i = 1 to n'l do
3 |   Select a random set of 5 variables
4 |   Select a random function (g_a, g_b) from the seeding set of bi-objective functions
5 |   Substitute cells corresponding to the selected variables in Q1 by g_a, and in Q2
  |     by g_b
6 end for
```

We designed five different datasets containing instances created using different criteria. The first three datasets include instances generated using O. The first step consists in selecting a set of seeding submatrices from O. Each type of instance is defined by the action this set is selected. Then, a common algorithm is invoked to generate the instances from the set. All instances used here have dimension $n = 100$.

- (T1) *HardInst*: Comprises problems with at least 7 non-dominated solutions and deceptive values above 24. 10 instances.
- (T3) *LessPopInst*: The set comprises the bi-objective problems in O that have at least 4 non-dominated solutions and the degree of deception is above 29. 9 instances.
- (T4) *ComposeInstHard*: Comprises problems with at least 7 non-dominated solutions. The 100 such problems in O with maximum degree of deception are selected. 10 instances.

The second dataset has been manually engineered to include typical cases of deception. The last dataset contains random instances generated according to the method proposed in [Liefooghe et al., 2014].

- (T2) *ArtTypeInst*: Each of the instances corresponds to a different problem manually defined. 5 instances.

- (T5,T6,T7) *RandomInst*: The sparsity of the matrices was respectively set to 0.8, 0.02, and 0.03. For each instance type, one instance was generated for each correlation value in $\{-0.5, 0, 0.5\}$. Therefore there were 3 instances for each type.

5.5 Experiments

In addition to the exploratory analysis to investigate the space of mUBQP deceptive functions presented in Section 5.3, we empirically evaluated the way in which the characteristics of the generated instances influences the behavior of MOEAs. We applied our MOEA/D variants presented in the previous chapters: MOEA/D-GA, MOEA/D-BACO, and MOEA/D-PBIL-c.

Here, our goal is not to focus on the comparison between the algorithms but evaluate the difficulty of the instances introduced in the paper in a wider context, considering MOEAs with different search strategies.

The modification of the *IGD* known as IGD_p [Schütze et al., 2012] is used to asses the quality of the approximated *PFs* obtained by each algorithm. Schutze et al. demonstrated that IGD_p is more Pareto compliant than the original IGD.

We will measure the difficulty of the instances in terms of the ability of the algorithms to optimize this metric and the number of generations needed to produce a low IGD_p. Therefore, as a preliminary step and in order to compute the true *PF*, we have exhaustively executed the algorithms to produce a reference *PF* (P^*) for each instance. The algorithms were guaranteed a large number of function evaluations (5000 generations) to find accurate PF approximations. In a second step of the experiments, we have used the produced reference *PFs* to compute the IGD_p metric and size of the *PF* in each generation of the algorithms. From this information we could determine whether the IGD_p metric was minimized at the end of the search and the number of generations needed for minimizing it.

5.5.1 Parameters setting

The following parameters settings were used in the experimental study:

1. *Number of subproblems* (N)*:* The number of subproblems N and their correspondent weight vectors $\lambda^1, \ldots, \lambda^N$ are set according to the parameter $H = 100$ (described in Section 2.6), consequently $N = 101$.

2. *Selection and replacement neighborhood size*: The number of the selected solution S for each $B(i)$ is set to $0.2 \times N$, i.e., $T = 20$.

3. *Maximal number of replacements by a new solution* (n_r)*:* Each new generated solution $\mathbf{y}$ can update a maximum $n_r = 2$ parents solution from $B(i)$.

4. *Genetic operators from MOEA/D-GA: uniform crossover* and *mutation rate* $\mu = \frac{1}{n}$.

5. *Learning rate:* -PBIL-c $\alpha = 0.01$, and -BACO $\alpha = 0.1$.

6. *Number of generations (second phase):* $MaxGen = 500$.

For each of the 43 instances included in our study, the three algorithms performed 30 independent runs.

Table 5.2: Size of the true Pareto fronts for the test instances. Instances for which $q_{ij}^k \notin \{\pm 1\}$ are underlined.

index	T1	T2	T3	T4	T5	T6	T7
1	121	101	80	69	500	43	50
2	121	4921	80	72	289	22	23
3	121	243	80	67	140	13	11
4	121	9157	80	71			
5	121	174	80	70			
6	121		80	65			
7	141		60	67			
8	121		60	70			
9	121		60	59			
10	121			69			

5.5.2 Analysis of the true PFs

Table 5.2 shows the size of the *PFs* for all the instances. One of the findings from the analysis of the *PFs* was the observation that the most important factor in the size of *PFs* is the range of values from which the mUBQP instances are generated. The higher the range of the interval from which values of q_{ij}^k are sampled the wider the range of the output values of the functions and the *PF*. This is an issue that is usually overlooked. The *PF* can be arbitrarily enlarged by increasing the range of values in the matrices, but this does not necessarily imply a more "interesting" or difficult PF. In Table 5.2, instances for which q_{ij}^k were selected for a range different than $\{\pm 1\}$ are underlined. In the following, we focus on functions for which $q_{ij}^k \in \{\pm 1\}$.

5.5.3 Influence of the instance difficulty in the behavior of the algorithms

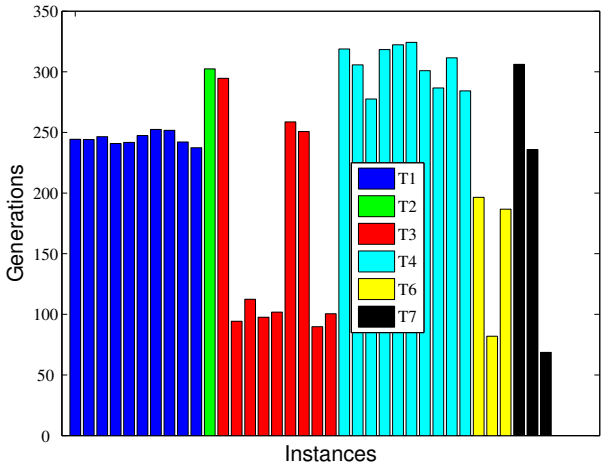
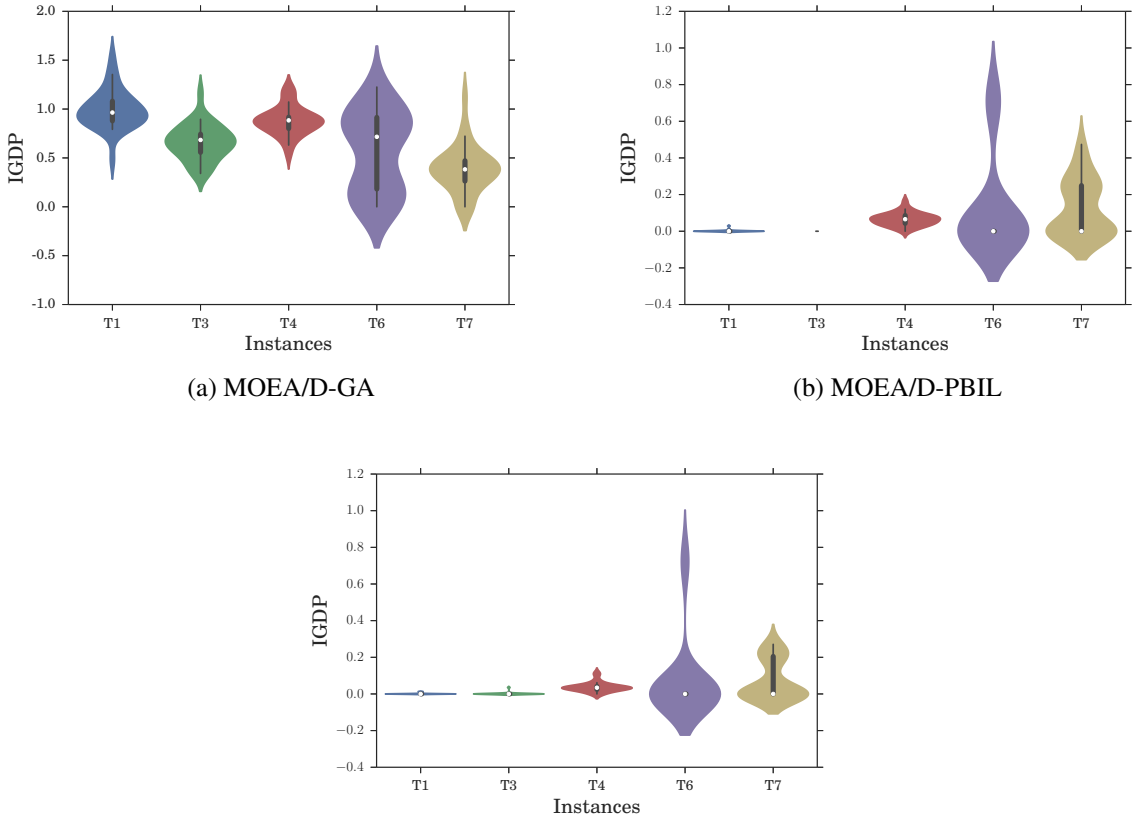


Figure 5.4: Average number of generations needed to reach an optimal IGD_p value for all instances.

We computed average number of generations needed to achieve an optimal value of the IGD_p. When the optimal value of IGD_p is not reached in a run we assign a value higher than

the number of generations. Figures 5.4 shows the average number of generations reached for all the instances. The values shown are an average of the results for all the algorithms, i.e. 90 runs.

It can be seen in Figure 5.4 that the number of generations to find reference IGD_p value changes considerably among the different classes of instances. All instances *ComposeInstHard* (T4) and *HardInst* (T1) required on average between 50 and 200 more generations than the random instances with the same sparsity (T6). As expected, increasing the number of non-zero values in the matrices (T7) allows having more populated PFs and the algorithms need more generations to obtain good approximations.



(a) MOEA/D-GA

(b) MOEA/D-PBIL

(c) MOEA/D-BACO

Figure 5.5: Distribution of the IGD_p metric achieved by MOEA/D variants for instances in classes $\{T_1, T_3, T_4, T_6, T_7\}$

Another potential use of instances is to investigate the behavior of different algorithms. We compare the behavior of the MOEA/D-GA, the MOEA/D-PBIL, and the MOEA/D-BACO in all instances in classes $\{T_1, T_3, T_4, T_6, T_7\}$. Figure 5.5 shows the average distribution of the IGD_p values for all the algorithms. To compute the distribution, we have first created a set containing IGD_p values for all instances in each class. Then a violin plot is created for each class showing the distribution of the values.

It can be seen in Figure 5.5 that the MOEA/D variants clearly exhibit different behaviors. Overall, MOEA/D-BACO and MOEA/D-PBIL have less difference between them compared to MOEA/D-GA. For the class $T1$ and $T3$, MOEA/D-BACO and MOEA/D-PBIL achieve the best results (low range of distribution), which means that they have achieved the reference IGD_p more frequently. MOEA/D-GA achieves the worst results for all classes of instances. A more

deep analysis of the difference between the models within MOEA/D to solve deceptive MCOPs is present in the next chapter.

5.6 Conclusions and future work

In this chapter, we have proposed a new method to generate hard instances for the mUBQP problem. Our algorithm is based on planting deceptive subsolutions in the matrices that define the mUBQP problem. We have empirically investigated the difficulty of the instances using different variants of the MOEA/D algorithm that add probabilistic-based operators. This study throw light on a number of issues that influence the complexity of mUBQP instances and that should be taken into account at the time of using this problem as a MOEA benchmark.

Our experimental results suggest that the introduced instances are indeed harder for MOEA/D than instances that are randomly generated. However, a more deep analysis is needed using other metrics to confirm that these instances are harder to solve than those from [Liefooghe et al., 2014].

Next chapter, we attempt to explore more complex EDAs in the context of MOEA/D for solving deceptive MCOPs.

Chapter 6

MOEA/D-GM: Using probabilistic graphical models in MOEA/D for solving combinatorial optimization problems

6.1 Introduction

Going more deeply on EDAs, we have extended our MOEA/D framework by incorporating multi-variate probabilistic models to solve complex MCOPs.

A PGM in the context of EDAs [Larrañaga et al., 2012] comprises a graphical component representing the conditional dependencies between the variables of the problem via tables of marginal and conditional probabilities. Additionally, the analysis of the graphical components of the PGMs learned during the search can provide information about the problem structure. PGMs within Multi-objective EDAs (MOEDAs) have been also applied for solving different MOPs [Pelikan et al., 2006, Karshenas et al., 2014].

For the experimental study present in this chapter, we have instantiated four MOEA/D variants, and they are called (i) MOEA/D-GA (which applies the genetic operators), (ii) MOEA/D-UMDA (using the marginal distribution probability), (iii) MOEA/D-PBIL-OS-c (presented in Chapter 4) and the novel (iv) MOEA/D-Tree (which comprises the multivariate Tree-EDA described in Section 2.4). We have investigated their search behaviors for solving the $bi\text{-}Trap_5(\mathbf{x})$, and the mUBQP instances from [Liefooghe et al., 2014] and those proposed in the previous chapter.

This chapter is organized as follows. Section 6.2 introduces the MOEA/D-GM. The related work is discussed in detail in Section 6.3. The experiments are described in Section 6.5. The conclusions and some trends for future work are presented in Section 6.7.

6.2 MOEA/D-GM: A MOEA/D framework with graphical models

In MOEA/D framework, a simple approach to introduce probabilistic models is *learning* and *sampling* a probabilistic model for each scalar subproblem i using the set of closest neighbors as the selected population. Therefore, at each generation, the MOEA/D-GM keeps N probabilistic models, and each model learned samples one solution.

In MOEA/D-GM, the model has to be set a priori. The framework follows the main MOEA/D guidelines presented in Section 2.6. In the initialization step, N solutions (population)

are initialized randomly, and the external Pareto (*EP*) is initialized with the non-dominated solutions of the initial population. Within the main while-loop, in case the termination criteria are not met, for each subproblem i, a probabilistic (graphical) model is learned using $B(i)$ as the selected population. Then, the sampling procedure is used to generate a new solution $\mathbf{y}$ from the respective model. The new solutions is used to update the parent population as in MOEA/D. A new solution can update a maximum of n_r current solutions. Finally, *EP* is updated with the non-dominated solutions from *Pop*.

Besides, the framework allows the application of the genetic operators using the same structure of *learning* and *sampling*. In the *learning step*, two parent solutions are selected from $B(i)$. The *sampling step* proceed with the *crossover* and *mutation*.

Let $i \in \{1, ..., N\}$, p^i represents a vector of positive distributions, and $p^i_{\mathcal{T}}$ represents the structure (e.g., matrix) of conditional probabilities in accordance to Equation (2.5) from Section 2.4.

Therefore, depending on the model of choice, the MOEA/D-GM maintains univariate probabilistic vectors p^i and/or matrices of conditional probability distributions $p\tau^i$ that is conformal with the Tree-EDA [Santana et al., 2001b]. Additionally, different classes of PGMs can be used for each scalar subproblem but we do not consider this particular scenario in this analysis.

We have already compared MOEA/D-BACO and MOEA/D-PBIL-OS-c in Chapter 4. Therefore, we have not incorporated the -BACO in this study. Additionally, MOEA/D-GM also includes two design modifications already introduced in our approaches. In the following, we briefly mention them.

Optimal Mutation Rate for EDAs: We have also used the Bayesian *priors* [Mahnig e Mühlenbein, 2001] to introduce a mutation operator. As mention before, the Bayesian priors is used in such a way that the computed probabilities will include a mutation-like effect [Mahnig e Mühlenbein, 2001].

Diversity preserving sampling (ds): In preliminary experiments, we have detected that one cause for early convergence of the algorithm was that solutions that were already in the population were newly sampled. As a form to avoid this situation, we have also used the simple procedure in which each new sampled solution is tested for presence in the neighborhood $B(i)$.

6.3 Related work

In this section, we review a number of related works emphasizing the differences to the work presented in this chapter.

6.3.1 MOEA/D using univariate EDAs

In [Zhou et al., 2013a], the multi-objective estimation of distribution algorithm based on decomposition (MEDA/D) is proposed for solving the multi-objective Traveling Salesman Problem (mTSP). For each subproblem i, MEDA/D uses a matrix Q^i to represent the connection "strength" between cities in the best solutions found so far. Matrix Q^i is combined with a priori information about the distances between cities of problem i, in a new matrix P^i that represents the probability that the sth and tth cities are connected in the route of the ith sub-problem. Although each matrix P^i encodes a set of probabilities relevant for each corresponding TSP subproblem, these matrices can not be considered PGMs since they do not comprise a graphical component representing the dependencies of the problem. The type of updates applied to the matrices is more related to parametric learning than to structural learning as done in PGMs. Furthermore,

this type of "models" resemble more the class of structures traditionally used for ACO and they heavily depend on the use of prior information (in the case of MEDA/D, the incorporation of information about the distances between cities). Therefore, we do not consider MEDA/D as a member of the MOEA/D-GM class of algorithms.

Another approach combining the use of probabilistic models and MOEAD for mTSP is presented in [Shim et al., 2012]. In that paper, a univariate model is used to encode the probabilities of each city of the TSP being assigned to each of the possible positions of the permutation. Therefore, the model is represented as a matrix of dimension $n \times n$ comprising the univariate probabilities of the city configurations for each position. One main difference of this hybrid MEDA/D with our approach, and with the proposal of Zhou et al [Zhou et al., 2013a], is that a single matrix is learned using all the solutions. Therefore, the information contained in the univariate model combines information from all the subproblems, disregarding the potential regularities contained in the local neighborhoods. Furthermore, since the sampling process from the matrix does not take into account the constraints related to the permutations, repair mechanisms and penalty functions are used to "correct" the infeasible solutions. As a consequence, much of the information sampled from the model to the solution can be modified by the application of the repair mechanism.

These previous MEDA/Ds were applied for solving permutation-based MOPs. The application of EDAs for solving permutation problems is increasing in interest [Ceberio et al., 2012]. Also, a number of EDAs specifically designed to deal with permutation-based problems have been proposed [Ceberio et al., 2011, Ceberio et al., 2013]. The framework proposed in this chapter is only investigated for binary problems. We present the review of EDAs specially tailored for permutation space in Chapter 7.

In [Wang et al., 2015], the authors proposed a univariate MEDA/D for solving the multi-objective knapsack problem (MOKP) that uses an adaptive operator at the sampling step to preserve diversity, i.e., prevents the learned probability vector from premature convergence. Therefore, the sampling step depends on both the univariate probabilistic vector and an extra parameter "r".

6.3.2 MOEA/D using multivariate EDAs

In [Giagkiozis et al., 2014], a decomposition-based algorithm is proposed to solve many-objective optimization problems. The proposed framework (MACE-gD) involves two ingredients: (i) a concept called generalized decomposition, in which the decision maker can guide the underlying search algorithm toward specific regions of interest, or the entire Pareto front and (ii) an EDA based on low-order statistics, namely the cross-entropy method [Botev et al., 2013]. MACE-gD is applied on a set of many-objective continuous functions. The obtained results showed that the proposed algorithm is competitive with the standard MOEA/D and RM-MEDA [Zhang et al., 2008]. The class of low-order statistics used by MACE-gD (Normal univariate models) limit the ability of the algorithm to capture and represent interactions between the variables. The univariate densities are updated using an updating rule as the one originally proposed for the PBIL algorithm [Baluja, 1994].

In [Zapotecas-Martínez et al., 2015], the covariance matrix adaptation evolution strategy (CMA-ES) [Hansen e Ostermeier, 1996] is used as the probabilistic model of MOEA/D. Although CMA-ES was introduced and has been developed in the context of evolutionary strategies, it learns a Gaussian model of the search. The covariance matrix learned by CMA-ES is able to capture dependencies between the variables. However, the nature of probabilistic modeling in the continuous domain is different to the one in the discrete domain. The methods used for learning

and sampling the models are different. Furthermore, the authors state that their main purpose was *to investigate to what extent the CMA-ES could be appropriately integrated in MOEA/D and what are the benefits one could obtain*. Therefore, emphasis was put on the particular adaptations needed by CMA-ES to efficiently learn and sample its model in this different context. Since these adaptations are essentially different that the ones required by the discrete EDAs used in this chapter, the contributions are different.

6.3.3 Other MOEDAs

Pelikan et al. [Pelikan et al., 2006] discussed the multi-objective decomposable problems and their difficulty. The authors attempted to review a number of MOEDAs, such as: multi-objective mixture-based iterated density estimation algorithm (mMIDEA) [Thierens e Bosman, 2001], multi-objective mixed Bayesian optimization algorithm (mmBOA) [Laumanns e Ocenasek, 2002] and multi-objective hierarchical BOA (mohBOA) [Khan, 2003]. Moreover, the authors introduced an improvement to mohBOA which combines three ingredients: (i) the hierarchical Bayesian optimization algorithm (hBOA), (ii) the multi-objective concepts from NSGAII [Deb et al., 2002] and (iii) clustering in the objective space. The experimental study showed that the mohBOA efficiently solved multi-objective decomposable problems with a large number of competing *building blocks*. The algorithm was capable of effective recombination by building and sampling Bayesian networks with decision trees, and significantly outperformed algorithms with standard variation operators on problems that require effective linkage learning.

All MOEDAs covered in [Pelikan et al., 2006] are Pareto-based and they use concepts from algorithms such as NSGAII and SPEA2. Since, in the past few years, the MOEA/D framework has been one of the major frameworks to design MOEAs. Incorporating probabilistic graphical models into MOEA/D seems to be a promising technique to solve scalable deceptive multi-objective problems.

Martins et al. [Martins et al., 2011] proposed a new approach for solving decomposable deceptive multi-objective problems. The MOEDA, called moϕGA, uses a probabilistic model based on a phylogenetic tree. The moϕGA was tested on the multi-objective deceptive functions f_{trap5} and f_{inv_trap5}. The moϕGA outperformed mohBOA in terms of number of function evaluations to achieve the exact PF, specially when the problem in question increased in size. A question discussed in [Martins et al., 2011] is: *if such probabilistic model can identify the correct correlation between the variables of a problem, the combination of improbable values of variables can be avoided. However, as the model becomes more expressive, the computational cost of incurred by the algorithm to build the model also grows. Thus, there is a trade-off between the efficiency of the algorithm for building models and its accuracy.*

In our framework, at each generation, N probabilistic graphical models are kept. Therefore, a higher number of subproblems and a large problem-scale can be a drawback in terms of efficiency (computational cost). However, if it has an adequate commitment between the efficiency and accuracy (quality of the outcomes), then it is expected to behave satisfactorily.

6.3.4 Contributions with respect to previous work

We summarize some of the main contributions of our work with respect to the related research.

- We use, for the first time, a probabilistic graphical model within MOEA/D to solve MCOPs. In this case, the previous MOEA/Ds that incorporate probabilistic models cover only univariate models.

- We investigate a particular class of problems (deceptive MOPs) for which there exist extensive evidence about the convenience of using probabilistic graphical models.
- We investigate the question of how the problem interactions are kept in the scalar subproblems and how these interactions are translated to the probabilistic models.

6.4 Multi-objective *Trap-k*:

There exists a class of scalable problems where the difficulty is given by the interactions that arise among subsets of decision variables. Thus, some problems should require the algorithm to be capable of linkage learning, i.e., identifying and exploring interactions between the decision variables to provide effective exploration. Decomposable deceptive problems have played a fundamental role in the analysis of EAs [Pelikan et al., 2007, Larrañaga et al., 2012]. One example of this class of decomposable deceptive functions is the *Trap-k*, where k is the fixed number of variables in the subsets (also called, partitions or *building blocks*) [Deb e Goldberg, 1993]. Trap deceives the algorithm away from the optimum if interactions between the variables in each partition are not considered. According to [Pelikan et al., 2005], that is why standard crossover operators of genetic algorithms fail to solve traps unless the bits in each partition are located close to each other in the chosen representation.

Pelikan et al. [Pelikan et al., 2005] used a bi-objective version of *Trap-k* for analyzing the behavior of multi-objective hierarchical BOA (mhBOA). The functions f_{trap5} (Equation (6.1)) and f_{inv_trap5} (Equation (6.2)) consist in evaluating a vector of decision variables $\mathbf{x} \in \{0,1\}^n$, in which the positions of $\mathbf{x}$ are divided into disjoint subsets or partitions of 5 bits each (n is assumed to be a multiple of 5). The partition is fixed during the entire optimization run, but the algorithm is not given information about the partitioning in advance. Bits in each partition contribute to *trap* of order 5 using the following functions:

$$f_{trap5}(\mathbf{x}) = \sum_{k=0}^{l-1} \sum_{i=1}^{5} trap5(x_{5*k+i}) \tag{6.1}$$

$$trap5(u) = \begin{cases} 5 & \text{if } u = 5 \\ 4-u & \text{if } u < 5 \end{cases}$$

$$f_{inv\_trap5}(\mathbf{x}) = \sum_{k=0}^{l-1} \sum_{i=1}^{5} inv\_trap5(x_{5*k+i}) \tag{6.2}$$

$$inv\_trap5(u) = \begin{cases} 5 & \text{if } u = 0 \\ u-1 & \text{if } u > 0 \end{cases}$$

where l is the number of building blocks, i.e., $n = 5l$, and u is the number of ones in the input string of 5 bits.

In the bi-objective problem, f_{trap5} and f_{inv_trap5} are conflicting. Furthermore, the amount of possible solutions grows exponentially with the problem size [Sastry et al., 2005].

6.5 Experimental study

First, we present the analysis regarding the bi-objective deceptive function *Trap-k*. The $bi\text{-}Trap_5(\mathbf{x})$ is defined as follows:

$$bi\text{-}Trap(\mathbf{x}) = (f_{trap5}(\mathbf{x}), f_{inv\_trap5}(\mathbf{x})) \quad subject\ to\ \mathbf{x} \in \{0,1\}^n \tag{6.3}$$

We have evaluated it with three different problem-sizes $n \in \{30, 50, 100\}$.

6.5.1 Parameters Settings

In the following, we describe the parameters settings.

1. *Number of subproblems* (N)*:* As in most of MOEA/D algorithms proposed in the literature [Zhang e Li, 2007, Li e Zhang, 2009, Deb e Jain, 2014], the number of subproblems N and their correspondent weight vectors $\lambda^1, \ldots, \lambda^N$ are controlled by a parameter H. For the bi-objective problem, we set $H = 200$, consequently $N = 201$.

2. *Neighborhood size* (T) *for selection and replacement*: As the number of selected solutions is crucial for EDAs, we test a range of neighborhood size values $T = (1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150)$.

3. *Maximal number of replacements by a new solution:* $n_r = 2$.

4. *Scalarization function:* We have applied both *Weighted Sum* and *Tchebycheff* approaches. As both achieve very similar results, only the results with *Tchebycheff* are presented in this chapter.

5. *Genetic operators from MOEA/D-GA: uniform crossover* and *mutation rate* $\mu = \frac{1}{n}$.

6. *PBIL learning rate:* $\alpha = 0.05$.

Each combination of (algorithm × parameters setting × problem-scale) is independently run 30 times.

6.5.2 Comparison results

For the *bi-Trap-k* the true *PF* is known. Thus, we have used the IGD indicator. We also present the cardinality of the outcomes. First, the algorithms are evaluated using a fixed neighborhood size ($T = 20$) and with and without the diversity preserving mechanism.

Table 6.1 presents the average values of the IGD values and the number of *true* Pareto solutions $|P^+|$ computed using found by the algorithms. Table 6.2 presents the results of the *Kruskall-Wallis* test (at 5% significance level) applied to the results obtained (the same results summarized in Table 6.1). The best rank algorithm(s) is(are) highlighted in boldface. Figure 6.1 and 6.2 show the behavior of the algorithms throughout the generations according to the average $|P^+|$ obtained.

From the analysis of the results, we can extract the following conclusions:

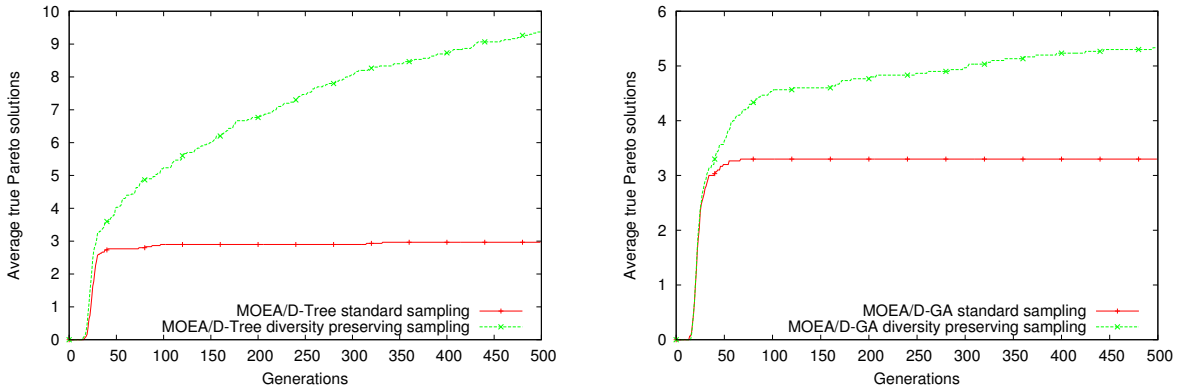
According to Table 6.2, the MOEA/D-Tree that uses the diversity preserving mechanism (MOEA/D-Tree-ds) is the only algorithm that has achieved the best rank in all the cases according

Table 6.1: Results from the average IGD and average number of *true* Pareto optimal solutions $|P^+|$ computed form the 30 runs for each combination (problem size $n\times$ Algorithm $\times$ preserving sampling). The column $|P^*|$ is the total number of *true* Pareto solutions for each problem n.

Average number of true Pareto optimal solutions $\|P^+\|$									
n	$\|P^*\|$	standard sampling				diversity preserving sampling (ds)			
		MOEA/D-GA	MOEA/D-PBIL	MOEA/D-UMDA	MOEA/D-Tree	MOEA/D-GA	MOEA/D-PBIL	MOEA/D-UMDA	MOEA/D-Tree
30	7	4.167	3.034	2.767	6.067	6.267	4.034	4.134	**6.9**
50	11	3.867	2.567	3.034	5.134	6.634	2.867	3.2	**9.5**
100	21	3.3	2.534	3.5	2.967	5.334	2.8	3.433	**9.367**
average IGD measure $IGD(P^*, P)$									
30	7	1.076	1.764	1.853	0.418	0.371	1.258	1.246	**0.053**
50	11	2.174	3.585	3.229	1.657	1.308	3.4	3.158	**0.501**
100	21	4.757	7.663	6.38	4.815	4.142	7.473	6.726	**3.359**

Table 6.2: Kruskall Wallis statistical ranking test according to the IGD and $|P^+|$ for each combination (problem size $n\times$ Algorithm $\times$ preserving sampling). The first value is the average rank over the 240 runs (i.e., 30 runs $\times$ 8 algorithms). The second value (in brackets) is the final ranking from 8 to 1. If two or more ranks are the same, it means that there is no significant difference between them.

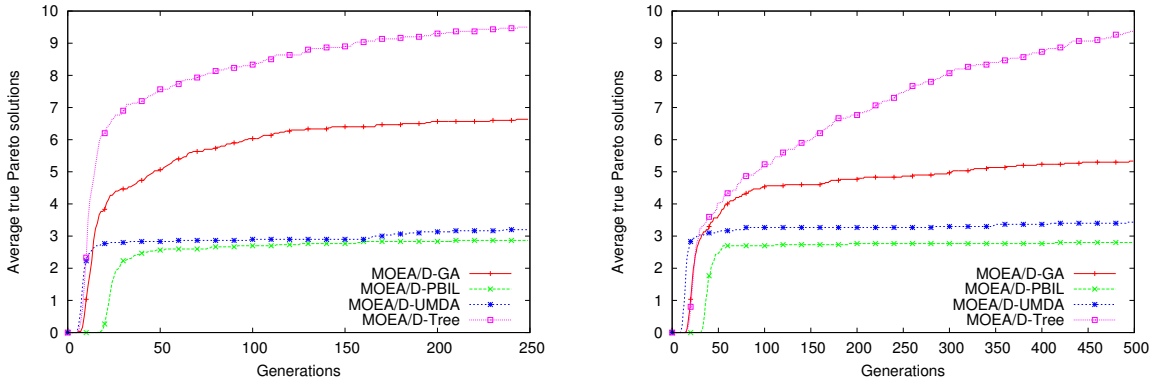
Kruskall-Wallis ranking - number of true Pareto optimal solutions $\|P^+\|$								
n	standard sampling				diversity preserving sampling (ds)			
	MOEA/D-GA	MOEA/D-PBIL	MOEA/D-UMDA	MOEA/D-Tree	MOEA/D-GA	MOEA/D-PBIL	MOEA/D-UMDA	MOEA/D-Tree
30	139.37 (5.50)	184.88 (6.00)	195.72 (6.50)	**66.25 (2.00)**	**59.43 (2.00)**	146.68 (6.00)	141.87 (6.00)	**29.80 (2.00)**
50	126.48 (5.00)	188.30 (6.50)	165.55 (6.00)	88.15 (3.00)	52.65 (2.00)	171.40 (6.00)	153.43 (6.00)	**18.03 (1.50)**
100	136.43 (5.50)	178.67 (6.00)	120.37 (5.00)	151.77 (5.50)	61.58 (2.00)	162.40 (5.50)	128.37 (5.50)	**24.42 (1.50)**
Kruskall-Wallis ranking - IGD measure								
30	134.13 (5.50)	187.05 (6.00)	195.22 (6.50)	**65.32 (2.00)**	**59.43 (2.00)**	148.67 (6.00)	144.38 (6.00)	**29.80 (2.00)**
50	106.88 (3.50)	194.43 (6.50)	169.37 (6.50)	74.47 (2.50)	58.33 (2.50)	179.35 (6.50)	160.17 (6.00)	**21.00 (2.00)**
100	**80.30 (2.50)**	204.03 (7.00)	145.72 (6.00)	**82.08 (2.50)**	**55.45 (2.50)**	195.37 (6.50)	162.12 (6.50)	**38.93 (2.50)**



(a) MOEA/D-Tree (b) MOEA/D-GA

Figure 6.1: Average *true* Pareto optimal solutions $|P^+|$ over the generations. Each chart shows a comparison between the standard sampling and the diversity preserving sampling using the same algorithm.

to both indicators. Besides, the diversity preserving sampling has improved the behavior of all the algorithms. Figure 6.1 confirms these results. Therefore, the diversity preserving sampling has a positive effect in the algorithms for the *bi-Trap5* problem, i.e., the algorithms are able to achieve a more diverse set of solutions. Additionally, Figure 6.2 shows that MOEA/D-Tree-ds achieves better results than the other algorithms since the first generations.

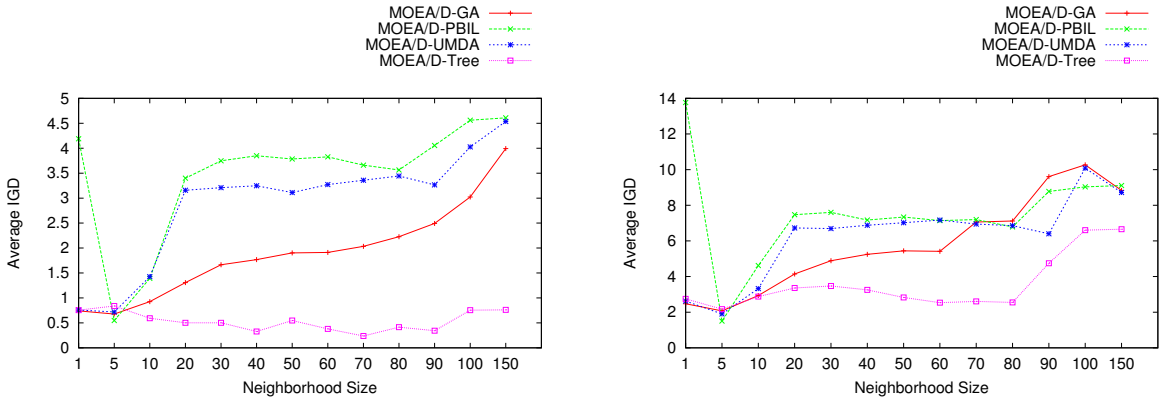


(a) $n = 50$ (b) $n = 100$

Figure 6.2: Average *true* Pareto solutions $|P^+|$ through the generations for problem size 50 and 100 with $T_s = 20$

All the algorithms can find, at least, one global optimal solution, i.e., a true Pareto solution. One possible explanation for this behavior is a potential advantageous effect introduced by the clustering of the solutions determined by the MOEA/D framework. This benefit would be independent of the type of models used to represent the solutions. Grouping similar solutions in a neighborhood may allow the univariate algorithm to produce some global optima, even if the functions are deceptive.

6.5.3 Influence of the neighborhood size selection T_s for the learning



(a) $n = 50$ (b) $n = 100$

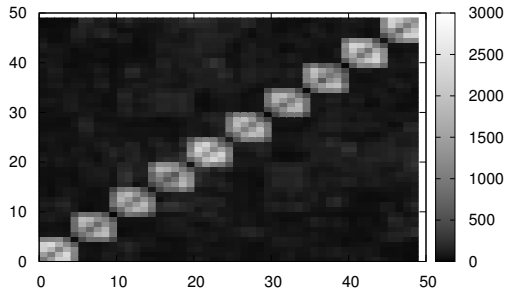
Figure 6.3: Average IGD with different neighborhood sizes for selection $T \in \{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150\}$

The neighborhood size has a direct impact in the search ability of the MOEA/Ds, which can balance convergence and diversity. Figure 6.3 presents the IGD values obtained with different neighborhood sizes (T).

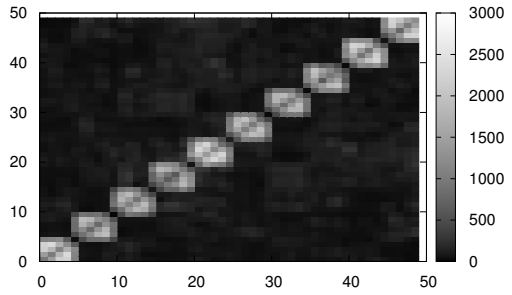
A multi-variate EDA needs a large set of selected solutions to be able to learn dependencies between the decision variables [Mühlenbein e Paass, 1996]. Our results confirm this assumption, which explain the differences between the algorithms in Figure 6.3. A remarkable point in these results is that, a small T is better to solve *bi-Trap5* for all algorithms except for

the MOEA/D-Tree. MOEA/D-Tree achieves good results with a large neighborhood size (e.g., 60, 70 and 80). Even if the neighborhood size has an important influence in the behavior of the algorithm, this parameter can not be seen in isolation of other parameters that also influence the behavior of the algorithm.

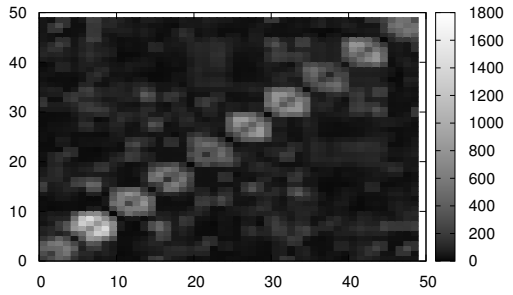
6.5.4 Analyzing the structure of problems as captured by the tree model



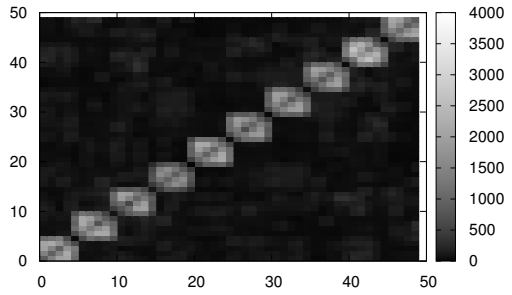
(a) $T = 20$, subproblem $i = 1$



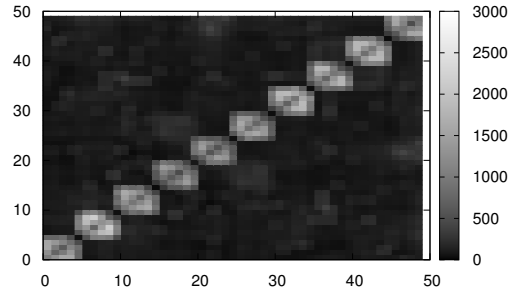
(b) $T = 70$, subproblem $i = 1$



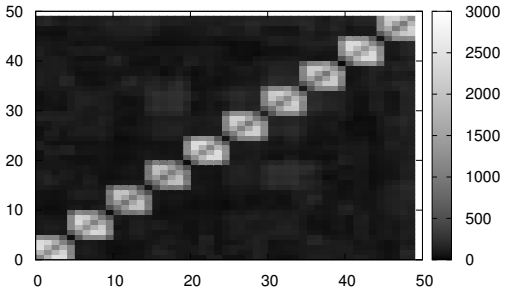
(c) $T = 20$, subproblem $i = 201$



(d) $T = 70$, subproblem $i = 201$



(e) $T = 20$, subproblem $i = 100$



(f) $T = 70$, subproblem $i = 100$

Figure 6.4: Frequency matrices (*heat maps*) learned by the subproblems $i = (1, 100, 201)$ from the MOEA/D-Tree-ds for problem size $n = 50$.

One of the main benefits of EDAs is their capacity to reveal a priori unknown information about the problem structure. Although this question has been extensively stud-

ied [Brownlee et al., 2012, Echegoyen et al., 2007, Santana et al., 2008] in the single-objective domain, the analysis in the multi-objective domain are still few [Fritsche et al., 2015, Karshenas et al., 2014, Santana et al., 2009]. Therefore, a relevant question was to determine to what extent is the structure of the problem captured by the probabilistic models used in MOEA/D-GM. This is a relevant question since there is no clue about the types of interactions that could be captured from models learned for scalarized functions. In this section, the structures learned from MOEA/D-Tree-ds while solving different subproblems are investigated.

In each generation, for each subproblem, a tree model is built according to the bi-variate probabilities obtained from its selected population. We can represent the tree model as a matrix $M_{n\times n}$, where each position M_{jk} represents a relationship (*pairwise*) between two variables j, k. $M_{jk} = 1$ if j is the parent of k in tree model learned, otherwise $M_{jk} = 0$.

Figure 6.4 represents the merge frequency matrices obtained by the 30 runs. The frequencies are represented using *heat maps*, where lighter colors indicate a higher frequency. We have plotted the merge matrices learned from the extreme subproblems and the middle subproblem i.e., $i = (1, 100, 201)$.

The matrices clearly show a strong relationship between the subsets of variables "*building blocks*" of size 5, which shows that the algorithm is able to learn the structure of the *Trap5* functions. Notice that, for neighborhood size 70, the MOEA/D-Tree-ds was able to capture more accurate structures, which exalts the good results found in accordance with the IGD metric. This can be explained by the fact that a higher population size reduces the number of spurious correlations learned from the data.

Moreover, analyzing the different scalar subproblems, we can see that, the algorithm is able to learn a structure even for the middle scalar subproblem $i = 100$, where the two conflict objectives functions f_{trap5} and f_{inv_trap5} compete in every partition (*building block*) of the decomposable problem.

6.6 Discussion: MOEA/D-GM to solve mUBQP

We have showed that a more complex probabilistic model that learns dependences between the variables is efficient to solve the *bi-Trap* function. Therefore, we have extended the analysis by evaluating the framework to solve the mUBQP instances. Our goal was to expand the analysis of the search behavior of MOEA/D-Tree to a wider class of deceptive MCOPs. However, in this scenario, we have faced some limitations.

Regarding, the general large-scale mUBQP instances from [Liefooghe et al., 2014], the main shortcoming is the high computational cost to learn and sample marginal and conditional probabilities, where the cost increases quadratically as the problem-scale increases. Therefore, the application of MOEA/D-Tree for large-scale instances, such as $n = 1000$, is inefficient. Additionally, as discussed in Chapter 5, the conventional mUBQP instances are generated randomly and they do not consider building blocks of deception. So, a univariate model is quite enough to solve the problem in this case.

Regarding the hard mUBQP instances (proposed in Chapter 5), which we have studied only to small-scale instances (e.g., $n = 100$), the results obtained by MOEA/D-Tree were not as expected. The results showed that, unfortunately, all the algorithms (MOEA/D-GA, MOEA/D-PBIL, MOEA/D-UMDA and MOEA/D-Tree) obtained similar results, i.e., the algorithms do not have a significance difference according to the *Hypervolume* indicator and the Kruskal-Wallis test (at 5% of significance level). Therefore, regarding the *trade-off* between the accuracy and the computational cost, MOEA/D-Tree is outperformed by the other algorithms to solve mUBQP.

6.7 Conclusions and future work

In this chapter, a novel and general MOEA/D framework able to instantiate probabilistic graphical models named MOEA/D-GM has been introduced. PGMs are used to obtain a more comprehensible representation of a search space. Consequently, the algorithms that incorporate PGMs can provide a model expressing the regularities of the problem structure as well as the final solutions. The PGM investigated in this chapter takes into account the interactions between the variables by learning a maximum weight spanning tree from the bi-variate probabilities distributions.

In terms of accuracy, the results show that the MOEA/D-Tree significantly outperforms the other MOEA/D variants for solving the *bi-Trap5* function. Furthermore, the results show that our diversity preserving mechanism introduced into the framework improves the algorithm search ability. An analysis of the influence of the neighborhood size on the behavior of the algorithms were conducted. In general, increasing the neighborhood size has a detrimental effect. Although, this is not always the case for the MOEA/D-Tree. Also, independent of the type of models used to represent the solutions, grouping similar solutions in a neighborhood may allow to produce some global optima, even if the functions are deceptive. The "most appropriate" model for *bi-Trap5* should be able to learn higher order interactions (of order 4).

Even if relatively small neighborhoods are used (in comparison with the standard population sizes used in EDAs), the models are able to capture the interactions of the problem. One potential application of this finding, is that, we could reuse or transfer models between subproblems, in a similar idea to the application of structural transfer between related problems [Santana et al., 2012].

The scalability of MOEA/D-Tree for the mUBQP was a limitation of this work. MOEA/D-Tree does not achieve a significant difference compared to the other variants to solve the mUBQP instances investigated.

In the future, other PGMs, for instance, based on Bayesian or Markov networks could investigated. Other directions for future work are: (i) Conceive strategies to avoid learning a model for each subproblem. This would improve the results in terms of computational cost; (ii) Use of the most probable configurations of the model to speed up convergence; (iii) Consider the application of hybrid schemes incorporating local search that take advantage of the information learned by the models; (iv) Evaluate MOEA/D-GM for solving other deceptive MCOPs.

Until here, we have worked with MCOPs with binary representations. However, several real-world problems can be represented as a permutation. Besides its useful application, permutation optimization problems have been used to provide algorithms challenges [Yenisey e Yagmahan, 2014]. The use of EDAs for solving permutation problems is considered recent [Ceberio et al., 2012]. The next chapter of this thesis attempts to propose a MOEA/D variant based on Mallows Models EDA [Ceberio et al., 2011] for solving permutation-based MOPs.

Chapter 7

Multi-objective Decomposition-based Mallows Models EDA. A case of study for Permutation Flowshop Scheduling Problem

7.1 Introduction

Following with our research on EDAs, we have investigated them for solving permutation-based optimization problems. This kind of problems reflect several real-world scenarios in different fields of application, such as engineering, computer science, finance, industry, etc. Behind the combinatorial nature of the solutions, every permutation-based problem are characterized with particular challenges that have to be faced [Ceberio et al., 2012].

Over time, several approaches (including exact methods, heuristics, and metaheuristics) have been proposed to deal with permutation-based optimization problems [Minella et al., 2011, Yenisey e Yagmahan, 2014]. Metaheuristics such as Simulated Annealing [Brooks e Morgan, 1995], Genetic Algorithms [Holland, 1975], Tabu Search [Glover, 1989] have shown their potentiality to solve permutation problems [Yenisey e Yagmahan, 2014].

Recently, Ceberio et. al. [Ceberio et al., 2011] proposed the use of a *distance-based exponential probability model* called Mallows Model (MM) [Mallows, 1957] in the context of EDAs to solve single-objective permutation-based optimization problems.

The MM is considered analogous to the Gaussian probability distribution over the space of permutations. In [Ceberio et al., 2014a] the authors proposed the use of the Generalized Mallows Model (GMM) in the context of EDAs (GM-EDA), and they achieved the state-of-the-art results for the permutation flowshop scheduling problem minimizing the total flow time. In addition, Kernels of Mallows models [Ceberio et al., 2015a] and Mixtures of Mallows models [Ceberio et al., 2015b] were proposed to deal with multi-modal problems.

While the application of probabilistic models defined on permutations has shown its potential for single-objective optimization problems, they have not been tested in the multi-objective case. Thus, in this chapter, we propose a multi-objective EDA based on decomposition and MM. The proposed framework involves two main ingredients: (i) Our MOEA/D framework, already presented in the previous chapters; and (ii) the use of kernels of Mallows models EDA [Ceberio et al., 2015a], which are able to model heterogeneous (multi-modal) populations. In Mallows Kernel EDA, a model is learned for every selected solution.

The main motivation behind combining these approaches is that, while the decomposition-based method deals with the search in different regions of the objective space, the learning and sampling steps of the kernels of Mallows models can efficiently produce promising solutions through the generations. It is worth noting that this is the first time that a MOEDA based on the use of a Mallows Model is proposed. The algorithm is called Multi-objective Decomposition-based Mallows Kernel EDA (MEDA/D-MK framework).

Additionally, in order to test the potentiality of MEDA/D-MK, we applied it to the multi-objective permutation flowshop scheduling problem (MoPFSP) minimizing the makespan and the total flow time as the multi-objective criteria. The problem has been proved to be *NP-hard*, and it is one of the most studied permutation optimization problems due to its significance in both academic and real-world application [Yenisey e Yagmahan, 2014].

Two components are incorporated into MEDA/D-MK for MoPFSP to enhance its performance: (i) the constructive $LR(n/m)$ heuristic algorithm [Liu e Reeves, 2001] to initialize the population, and (ii) a controlled perturbation procedure attempting to escape from local optima.

We have used the 110 benchmark instances from [Minella et al., 2011] for the experimental studies. We have evaluated MEDA/D-MK in two situations. First, we compare it to the MOEA/D (which applies genetic operators, specially tailored for minimizing makespan and total flowtime) [Chang et al., 2008, Alhindi e Zhang, 2014]. Next, we e have compared to a set of approximated (*PFs*) produced by the state-of-the-art algorithms from [Minella et al., 2011] [Dubois-Lacoste et al., 2011].

The remainder of the chapter is organized as follows. Section 7.2 introduces the proposed MEDA/D-MK framework and describes the Mallows model in detail. In Section 7.3, we present the target problem and the MEDA/D-MK algorithm configuration for MoPFSP. We present the experimental study in Section 7.4. Finally, in Section 7.5, we present the conclusions and future work.

7.2 The framework: Multi-objective decomposition-based Kernel of Mallows model EDA

In this section, we present one of the goals of this thesis: the proposal of a general multi-objective optimization algorithm based on the decomposition approach and the probabilistic models defined directly in the space of permutations (Mallows Models) for solving permutation-based MOPs efficiently. To the knowledge of the authors, this is the first time that an algorithm of this kind is proposed.

In the following, we describe the MEDA/D-MK framework. Subsequently, we present the Mallows model.

7.2.1 The main framework

Let σ denote a permutation vector solution and $\sigma(i)$ represent the ith position (variable) of the permutation. The general design follows our previous MOEA/D framework. Algorithm 7.1 presents the MEDA/D-MK pseudo-code. In the following, we describe its steps. In addition, Table 7.1 summarizes the parameters required by the algorithm.

1. **Initialization:** First, the N uniform distributed weight vectors $\lambda^1, .., \lambda^N$ are set as in Section 2.6. The initial population $Pop = (\sigma^1, ..., \sigma^N)$ is generated randomly or by a

Algorithm 7.1: MEDA/D-MK framework

```
1  Initialize N weight vectors, generate N permutation solutions Pop = {σ^1, ..., σ^N}
     randomly or by a problem-specific, and compute every F(σ^k)
2  Initialize EP with the non-dominated solutions from Pop
3  while a termination condition is not met do
4      for each subproblem k ∈ 1, ..., N do
5          σ_0^k, θ^k ← Estimate the MM parameters using B(k);
6          σ_s^k ← Sample a new solution from P^k(σ);
7          Compute F(σ_s^k):
8          Pop = Update_Neighborhood(σ_s^k);
9          If a condition is met, σ^k ← Shaking(σ^k)
10     end for
11     UpdateEP(Pop, EP)
12 end while
13 Return Pop, EP
```

problem-specific method. Then, their corresponding fitness functions $F(\sigma^1), ..., F(\sigma^N)$ are computed. The EP is initialized with the non-dominated solutions from the initial population Pop.

2. **Learning and Sampling:** The framework, at each generation, for each subproblem k, estimates the parameters θ^k and σ_0^k (see Section 7.2.2). Next, the parameters are used to compute the probability $P^k(\sigma)$ of each $\sigma \in \mathbb{S}_n$ according to the distance metric (Figure 7.1). Finally, it samples a new solution σ_s^k.

3. **Update the neighborhood:** Next, the sampled solution σ_s^k is used to update the current population as described in Section 2.6.

4. **Perturbation procedure:** A drawback of most approaches dealing with multi-objective permutation optimization problems is the lack of diversity, i.e., the search ability to explore different regions of the search space while moving towards the PF. Therefore, optionally, a destruction (perturbation) method can be performed in σ^k if it does not change after a maximum number of generations. This scheme is performed to escape from local optima, and thus to control the diversity of the population.

5. **Update the *EP*:** Finally, the EP is updated with the non-dominated solutions from Pop as described in Section 2.6.

When the stopping condition is met, the algorithm returns Pop and EP.

7.2.2 Mallows model

The Mallows model [Mallows, 1957] is a distance-based exponential probability model defined over permutation spaces. Under this model, the probability value of every permutation $\sigma \in \mathbb{S}_n$ (where $\mathbb{S}_n$ stands for the set of $n!$ permutations of n items) depends on two parameters: a spread parameter θ, and the distance to a central permutation σ_0, which is calculated by a distance-metric D. Formally, the Mallows model is defined as

$$P(\sigma) = \frac{e^{-\theta D(\sigma,\sigma_0)}}{\psi(\theta)} \tag{7.1}$$

Table 7.1: Summary of the input parameters for the Algorithm 2.4

Parameter	Description
H	Control parameter to generate the weight vectors as in [Zhang e Li, 2007]
N	Number of subproblems
λ^k	The weight vector associated to the subproblem k
T	Neighborhood size
θ	The spread parameter from Mallows models
σ_0	The central permutation from Mallows models
n_r	maximum replacements by a new solution

where $\psi(\theta)$ denotes the normalization constant.

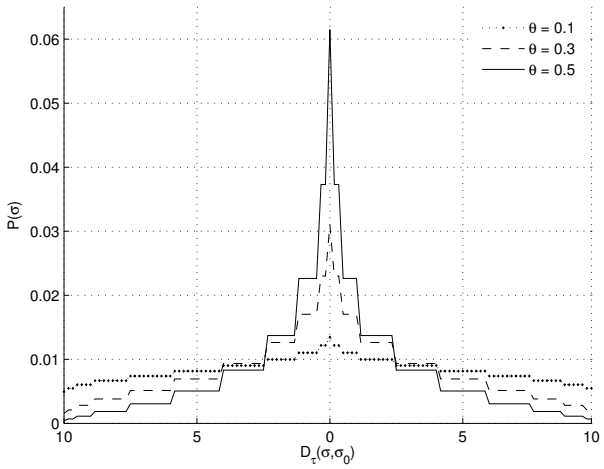


Figure 7.1: The Mallows model exponential probability distribution of a permutation σ with a spread parameter θ under a distance-metric D [Ceberio et al., 2011].

Figure 7.1 presents different exponential probabilities $P(\sigma)$ according to different spread parameter values (θ) and a distance $D(\sigma, \sigma_0)$. When $\theta > 0$, the central permutation σ_0 is the permutation with the highest probability value, and the probability of the other $n! - 1$ permutations exponentially decreases with the distance to σ_0. The larger the θ, the sharper the distribution around σ_0. When $\theta = 0$, a uniform distribution is obtained, i.e., the equation assigns equal probability to every permutation σ in $\mathbb{S}_n$.

The most common distance metrics used with the Mallows model are the Kendall's$_\tau$ distance [Fligner e Verducci, 1988, Ceberio et al., 2011] and the Cayley distance [Irurozki et al., 2014a]. Ceberio et al. [Ceberio et al., 2014b], reported that Cayley distance achieves the best results for the single-objective PFSP minimizing the total flow time. Therefore, we only describe the Cayley distance in this work. For additional information about the Kendall-τ and Ulam distance the reader is referred to [Ceberio et al., 2014b] and [Irurozki et al., 2014b] respectively.

The Cayley distance $D_c(\sigma, \pi)$, counts the minimum number of swaps (not necessary adjacent) that have to be performed to transform σ into π. When the reference permutation is the identity, $e = 123 \ldots n$, the number of swaps equals n minus the number of *cycles* of σ. A cycle is understood as a closed walk of elements in the permutation such that for every $1 \leq i, j \leq n$ for which $\sigma(i) = j$ is true, then i and j are in the same cycle. In addition, the Cayley

distance fulfills the *right-invariant* property, which means that $D(\sigma, \pi) = D(\sigma\gamma, \pi\gamma)$ for every permutation $\gamma \in \mathbb{S}_n$, where $\sigma\gamma$ stands for the composition of the permutations σ and γ, and is defined as $\sigma\gamma(i) = (\sigma \circ \gamma(i)) = \sigma(\gamma(i))$. Thus, by *right invariance*, when $\gamma = \pi^{-1}$, $D(\sigma, \pi)$ can be simplified as $D(\sigma\pi^{-1}, e)$ (also denoted as $D(\sigma\pi^{-1})$).

The Cayley distance $D_c(\sigma)$ can be decomposed as the sum of $n-1$ terms:

$$D_c(\sigma, \pi) = \sum_{j=1}^{n-1} X_j(\sigma\pi^{-1})$$

where $X_j(\sigma\pi^{-1}) = 0$ if j is the largest item in its cycle in $\sigma\pi^{-1}$, and 1 otherwise.

Under this distance, the normalization constant $\psi(\theta)$, is formalized as:

$$\psi(\theta) = \prod_{j=1}^{n-1} \psi_j(\theta) = \prod_{j=1}^{n-1} (n-j)\exp(-\theta) + 1 \tag{7.2}$$

For the incorporation of the MM as a probabilistic model into EDAs, it is necessary to define effective learning and sampling methods. In the following, the general methods for these steps are presented.

Learning:

Given a set of permutations, which works as a selected population, the learning process consists of estimating the consensus (central) permutation (σ_0) and the spread parameter (θ) for MM (or ($\boldsymbol{\theta}$) for GMM). Different methods have been reported to estimate σ_0 and θ in the context of EDAs [Ceberio et al., 2014a, Ceberio et al., 2014b, Ceberio et al., 2015b].

Usually, this process is approached via maximum likelihood estimation (MLE). However, the time required for an exact learning scales factorially with the number of variables [Ceberio et al., 2014a]. To deal with this shortcoming, Ceberio et al. [Ceberio et al., 2014b], carried out an approximated learning mechanism. The estimated consensus permutation (σ_0) is calculated as the *set median permutation*, which is the permutation that minimizes the sum of the distances to the rest of the permutations in the sample. Once σ_0 is estimated, the MLE for the spread parameter θ for MM (or $\boldsymbol{\theta}$ for GM) is computed. The expression for this parameter is obtained by equaling to zero the derivate of the likelihood.

$$X_j = \frac{j}{j + \exp(\theta_j)} \tag{7.3}$$

Ceberio et al. [Ceberio et al., 2014b] have solved this equation through the Newton-Raphson algorithm. Formally, after the learning step, the Mallows model under the Cayley distance can be defined as follows:

$$P(\sigma) = \psi(\theta)^{-1} \exp(\sum_{j=1}^{n-1} -\theta X_j(\sigma\sigma_0^{-1})) \tag{7.4}$$

for every $\sigma \in \mathbb{S}_n$.

Sampling:

The sampling step attempts to obtain new solutions from the model learned (Eq. 7.4) given by the probability of any σ according to the $n-1$ binary variables $X_1(\sigma), ..., X_{n-1}(\sigma)$ in which the distance is decomposed. The probability of each $X_j(\sigma)$ follows:

$$P(X_j(\sigma\sigma_0^{-1}) = 1) = \frac{(n-j)\exp(-\theta)}{\psi_j(\theta)} \tag{7.5}$$

Then, the sampling procedure follows a random method to generate a permutation σ given $X(\sigma)$ [Irurozki et al., 2014a]. Ceberio et al. [Ceberio et al., 2015a] described the sampling process according to three steps. First, a random binary vector $X(\sigma\sigma_0^{-1})$ is sampled using Eq. (7.5). Second, the associated $\sigma\sigma_0^{-1}$ is calculated according to the techniques described in [Irurozki et al., 2014a]. Finally, by *right invariance*, the final permutation σ is obtained. The complexity of this procedure is $O(n^2)$.

7.2.3 Kernels of Mallows Models

Despite the success of MM and GMM in the context of EDAs, they are unimodal models, i.e., they can provide accurate distributions when the population of the problem is *homogeneous* (low sparsity between the selected solutions). However, these models are not flexible enough to accurately model populations with solutions that are very sparse regarding the distance metric considered under the model. To overcome this drawback, mixtures of distance-based probability models have been used to deal with multi-modal problems [Ceberio et al., 2015a, Ceberio et al., 2015b] .

In [Ceberio et al., 2015a], Kernels of Mallows models EDA was proposed, where a kernel is defined for each selected solution, using this solution as its central permutation. The spread parameter θ is predefined to control the exploration/exploitation of the kernel. Then, new solutions are sampled from each kernel under a distance metric D.

This strategy is suitable for the multi-objective case, since most of the MOPs are multi-modal due to the number of Pareto optimal solutions that they may contain. Moreover, one of the primary goals of MOEAs is to maintain a diverse population for finding solutions that cover the entire true PF.

Therefore, in the MEDA/D-MK framework, each subproblem k is associated to a Mallows Kernel. Consequently, N Mallows Kernel models are kept at every generation. The interaction between the subproblems is kept by the update step.

7.3 A case of study: MEDA/D-MK for Multi-objective Permutation Flowshop Scheduling Problem

The PFSP is an attractive research area. It is not only a theoretical field of study but also interesting field of application in industry and other real-world scenarios [Minella et al., 2008, Yenisey e Yagmahan, 2014].

The objective of the PFSP is to find a permutation that optimizes a particular criterion. The PFSP schedules n jobs with given processing times on m machines, where the sequence of processing a job on all machines is identical and unidirectional for each job. The problem has the following assumptions [Yenisey e Yagmahan, 2014]: (i) All jobs are available at time zero, (ii) each job can only be processed on, at most, one machine at the same time, (iii) each machine can process only one job at a time, (iv) different jobs have the same processing order on all machines, and (v) the set-up times of the jobs on machines are sequence-independent and are included in the processing times.

A feasible permutation solution is denoted as $\sigma = (\sigma(1), \sigma(2), ..., \sigma(n))$ where $\sigma(i)$ represents the job to be processed in the ith position. Let p_{ij} denote the processing time for job i

on machine j. C_{ij} is the completion time of job i on machine j. The completion time of a job i in the last machine m is indicated indicated as C_{im}, and it is recursively calculated as follows:

$$\begin{aligned} C_{11} &= p_{11} \\ C_{i1} &= C_{(i-1)1} + p_{i,1},\ i = 2, ..., n \\ C_{1j} &= C_{1(j-1)} + p_{1,j},\ j = 2, ..., m \\ C_{im} &= max(C_{(i-1)j}, C_{i(j-1)}) + p_{ij},\ i = 2, ..., n\ and\ j = 2, ..., m \end{aligned} \tag{7.6}$$

Usually, C_{im} is also indicated as $C_i(\sigma)$. A scheduling problem can naturally involve multiple objectives to be optimized at the same time. Most literature referring to the PFSP focuses on a group of objectives based on the completion time, e.g., the minimization of the total flow time (TFT) and the minimization of the makespan (C_{max}). Other groups of objectives have also been studied, such as the objectives based on due dates (tardiness) [Minella et al., 2008, Yenisey e Yagmahan, 2014].

The makespan represents the time needed to process all the jobs. Minimizing makespan is related to the increase of throughput and machine utilization, and it is formulated as follows:

$$C_{max}(\sigma) = \max\{C_i(\sigma)\},\ \ i = 1, ..., n \tag{7.7}$$

The flow time of a job (F_i) is the time elapsed between its release time and its completion time. As the release time is always 0, therefore, $F_i(\sigma) = C_i(\sigma)$. The total flow time is formulated as follows:

$$TFT(\sigma) = \sum_{i=1}^{n} C_i(\sigma),\ \ i = 1, ..., n \tag{7.8}$$

The total tardiness is defined as follows:

$$TT(\sigma) = \sum_{i=1}^{n} Ti,\ \ i = 1, ..., n \tag{7.9}$$

where T_j is the tardiness of job i, defined as $T_j = max\{C_i(\sigma) - d_i\}$ and represents the delay in its completion with respect to its due date (d_i).

Thus, for instance, we can define a MoPFSP as:

$$\begin{aligned} minimize\ \ F(\sigma) &= (C_{max}(\sigma), TFT(\sigma)), \\ &subject\ to\ \sigma \in \Omega \end{aligned} \tag{7.10}$$

7.3.1 MoPFSP: Related Work

Minimizing two objectives:

Minella et al. [Minella et al., 2008] reviewed and evaluated a total of 23 algorithms including both flowshop-specific and general multi-objective optimization approaches on 110 benchmark test instances. These PFSP instances are those of Taillard [Taillard, 1993] with the addition of due dates (which are used for the total tardiness criterion). The best approaches that they identified were those based on simulated annealing and genetic local search. The benchmark is available at their web page repository http://soa.itit.es.

Several works have used this Taillard benchmark to evaluate their approaches [Minella et al., 2011, Li e Li, 2015]. Consequently, we have also used it to evaluate our proposal.

Minella et al. [Minella et al., 2011] introduced an efficient metaheuristic algorithm based on an Iterated Greedy technique. The algorithm called restarted iterated Pareto greedy (RIPG), combines (i) an efficient initialization of the population using the NEH heuristic [Nawaz et al., 1983], (ii) the management of the approximated Pareto front, and (iii) a specially tailored local search. In all computational tests, and performance indicators, the proposed RIPG yields better results (many times in a significant way) than other 5 re-implemented (or adapted) state-of-the-art methods for the following pairs of objectives: (i) makespan and total flowtime and (ii) makespan and total tardiness. Additionally, their results are available at the same repository containing the benchmark. As we will show later, we include their *best-known* results in our comparison study.

In the same year, Dubois-Lacoste et al. [Dubois-Lacoste et al., 2011] proposed a hybrid algorithm, called TP+PLS, which combines a two-phase local search and a Pareto local search. The authors tested it for different bi-objective combinations (makespan, total flowtime, total tardiness), showing a good performance. Their best-known results are also included in our comparison study.

The authors in [Yenisey e Yagmahan, 2014] presented a more recent review of the approaches designed to deal with the MoPFSP and the benchmarks available in the literature. They reviewed 86 articles and reported that the number of studies has been gradually increasing in time. Moreover, the most investigated metaheuristic methods have been genetic operators, Tabu Search, and Simulated Annealing. Besides, most of these algorithms use a PFSP-specific constructive method to initialize the population. The authors pointed out that one trend in this area is the combination of efficient heuristic and metaheuristic methods to generate new hybrid algorithms in order to take advantage of the different approaches. The framework proposed in this chapter follows this trend.

Algorithms based on the MOEA/D framework have already been applied to solve MoPFSP. Chang et al [Chang et al., 2008], proposed the application of a MOEA/D variant, using the two-point crossover and the insert-based moving mutation, specifically tailored for minimizing makespan and total flowtime. Their results showed that the algorithm outperformed the Pareto-based algorithms NSGAII and SPEA2 for the set of test instances proposed by Ishibuchi et al. [Ishibuchi et al., 2003]. Likewise, Alhindi and Zhang [Alhindi e Zhang, 2014] hybridized MOEA/D and Tabu Search (TS). Their primary motivation was to use TS to help MOEA/D escape from local optimal solutions. As the TS has a higher computational cost, it was only applied to good solutions. The results showed that MOEA/D-TS outperformed the MOEA/D for the test instances proposed by [Ishibuchi et al., 2003]. As we will present later, we include a MOEA/D variant based on the same genetic operators used by [Chang et al., 2008, Alhindi e Zhang, 2014] in our experimental study.

The authors in [Li e Li, 2015] proposed a decomposition-based multi-objective local search algorithm (MOLSD). First, they used an efficient PFSP heuristic procedure to initialize the population. Next, the algorithm executes a Pareto local search embedded with a *heavy* perturbation procedure, followed by a single insert-based local search and a restarted method to escape from local optima. The results reported showed that MOLSD outperformed some state-of-the-art algorithms, such as RIPG [Minella et al., 2011]. Unfortunately, their *best-known* approximated *PFs* are not available for comparison.

Minimizing three objectives:

The number of studies focusing on three objectives is much smaller than the studies on bi-objective [Yenisey e Yagmahan, 2014]. Ishibuchi and Murata [Ishubuchi e Murata, 1998] proposed the multi-objective genetic local search algorithm (MOGLS). The algorithm applies a

local search procedure to each new solution obtained by the genetic operators. The authors tested the algorithm to minimize the objectives makespan, total flowtime and total tardiness.

The authors in [Arroyo e Souza Pereira, 2011] proposed an GRASP algorithm specially designed to solve bi- and tri-objective PFSP. In case of tri-objective, the algorithm outperformed two improved MOGLS variants [Arroyo e Armentano, 2005].

7.3.2 MEDA/D-MK for MoPFSP

In order to demonstrate the validity of MEDA/D-MK, we have applied it to solve the MoPFSP minimizing the total flowtime and the makespan. The experimental study regarding three objectives is presented in Appendix A.

First, we have defined some ad-hoc algorithm components, and they are described in the following.

The objectives have different scales. Therefore, as in [Li e Li, 2015], we have used the *max-min* normalization to compute the scalarizing functions (e.g., *Weighted sum* and *Tchebycheff*). However, when the objective function values are normalized, $\mathbf{z}^*$ does not guarantee a lowest reference value. To deal with this issue, each z_l is multiplied (decreased) by a factor α. For the the bi-objective PFSP, we set $\alpha = 0.6$ in accordance to [Chang et al., 2008]. The adapted *Weighted sum* and *Tchebycheff* scalarizing functions are as follows:

Weighted Sum:

$$\begin{aligned} \textit{minimize} \quad & g^{ws}(\sigma|\boldsymbol{\lambda}, \mathbf{z}^*, \mathbf{w}^*) = \sum_{l=1}^{q} \lambda_l \frac{f_l(\sigma) - \alpha\ z_l^*}{w_l^* - z_l^*} \\ & \textit{subject to } \sigma \in \Omega \end{aligned} \tag{7.11}$$

Thebycheff:

$$\begin{aligned} \textit{minimize} \quad & g^{tc}(\sigma|\boldsymbol{\lambda}, \mathbf{z}^*, \mathbf{w}^*) = \max_{1 \leq l \leq q} \left\{ \lambda_l \frac{f_l(\sigma) - \alpha\ z_l^*}{w_l^* - z_l^*} \right\}; \\ & subject\ to\ \sigma \in \Omega \end{aligned} \tag{7.12}$$

where $\mathbf{z}^*$ is the reference point, i.e., the minimum (best) values found so far for each objective value, and $\mathbf{w}^*$ is the maximum (worst) values found so far for each objective value.

1. **Initialization step**: First, the $LR(n/m)$ procedure constructs a single solution σ_{lr}. Next, one subproblem is randomly chosen to be initialized with σ_{lr}. Second, the algorithm randomly chooses $(N/2) - 1$ subproblems and initializes them with a light perturbed σ_{lr} solution. The perturbation on σ_{lr} is based on $n/10$ insert-based moves. This strategy allows the initial population to maintain some characteristics of σ_{lr} and also can find other promising solutions. Next, the remaining $N/2$ subproblems are initialized randomly to guarantee that the convergence and some diversity is maintained at the initial Pop. Finally, the EP is initialized with the non-dominated solutions from Pop.

2. **Learning step**: The central permutation σ_0^k is set to its current solution σ^k. In [Ceberio et al., 2015a], to enhance the search ability of the model, an adaptive interval of values for θ is defined. In their strategy, at every generation, if the best solution was not improved, then θ was increased. Otherwise, θ was set to the lower bound value. However, in our preliminary experiments, this strategy did not achieve successful results.

Besides, a fixed θ value at every generation obtained good results. The experimental study to assign different probabilities is presented in Section 7.4.6.

3. **Sampling step:** The new solution σ_s^k is sampled according to Eq. (7.4) and the steps described in Section 7.2.2. Further, the sampling step has been enhanced by the addition of two procedures:

   (a) A single insert-based movement is applied to σ_s^k with a low probability.

   (b) The diversity preserve mechanism is proceed to avoid that the same permutation appears many times in the neighborhood. So, if the sampled solution $\sigma_s^k \in B(k)$ then, the algorithm discards σ_s^k, and tries to sample a new one until it samples a solution $\sigma_s^k \notin B(k)$ or reaches a maximum number of trials (T). If T is reached, the algorithm maintains the last sampled solution anyway.

4. **Update *Pop***: the new sampled solution is used to update Pop. The update procedure follows the scheme described in the general framework (Section 2.6).

5. **Shaking procedure:** To lead the algorithm to escape from local optima and control the diversity of the population, MEDA/D-MK is improved with a controlled shaking procedure that can be performed at any subproblem k. The procedure works as follows: If σ^k has not been enhanced (updated) after a predefined number of generations ($count$), σ^k is updated by receiving n_{sh} random insert-based movements, even if it computes a worse $F(\sigma^k)$. Then, the $count$ for a new perturbation in the subproblem k is reset to 0.

Regarding the input parameters, besides those summarized in Table 7.1, the MEDA/D-MK for MopFSP includes: 1) the factor α for the adapted *Weighted Sum* and *Tchebycheff*, 2) the maximum number of consecutive generations without an improvement in the subproblem k before to execute the shaking procedure ($count^k$), and 3) the number of random insert-based movements (n_{sh}) to perturb the solution.

7.4 Experimental studies

The goal of the experimental studies is to evaluate the performance of MEDA/D-MK for MoPFSP compared to the state-of-the-art approaches. In this analysis, we: (i) evaluate if, within the MOEA/D framework, using the Mallows Model is preferable to genetic operators, and (ii) check if our approach is competitive compared to the state-of-the-art results reported by the scientific community. Additional experiments are present in Appendix A.

7.4.1 Algorithms and the PFSP benchmark

Murata and Ishibuchi [Murata e Ishibuchi, 1994] evaluated various genetic operators for makespan and total flowtime. Their results showed that the two-point crossover and the insert-based mutation (see Figure 7.2) achieved the best results. Thus, as done in [Chang et al., 2008, Alhindi e Zhang, 2014] to the MOEA/D variants for MoPFSP, we have incorporated these operators in our MOEA/D instantiation. It is worth noting that, for a fair comparison, our MOEA/D variant also incorporates the constructive PFSP initialization and the shaking procedure.

The execution time analysis was performed on a PC with Intel Xeon E5-620 2.4 GHz processor and 12 GB memory.

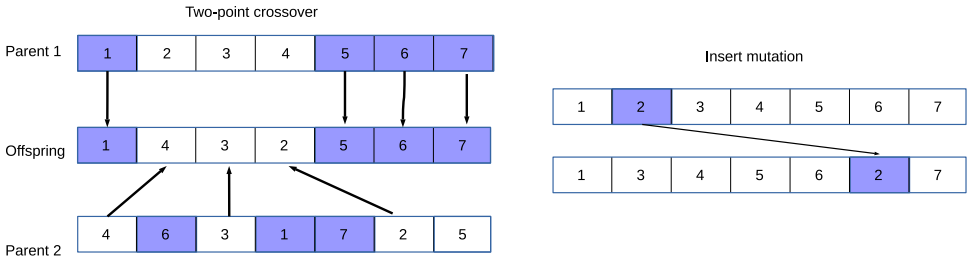


Figure 7.2: Two-point crossover and insert mutation operators

The Taillard PFSP benchmark [Minella et al., 2011] has been widely employed in the literature. Each processing time p_{ij} is generated from a random uniform distribution in the range $[0, 99]$. The benchmark is composed by 110 test instances having different combinations of number of jobs $n = \{20, 50, 100, 200\}$ and number of machines $m = \{5, 10, 20\}$. The instances are grouped in 11 different combinations (scale) $n \times m$ (20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10 and 200x20) containing 10 instances each.

Usually, if the source code is not available, the researchers have to re-implement the state-of-the-art algorithm(s) to compare them to their proposed approach. However, it is not easy to make a fair re-implementation, and sometimes they achieve different results regarding those previously reported. Another avenue is to use the *best-known* approximated *PFs* provided by the research community for comparison.

Usually, if the source code is not available, the researchers have to re-implement the state-of-the-art algorithm(s) to compare them to the proposed approach. However, it is not easy to make a fair re-implementation, and sometimes they achieve different results regarding those previously reported, as it was stated in [Minella et al., 2011]. The other option is to use the *best-known* approximated *PFs* provided by the research community for comparison.

7.4.2 Parameters Setting

In the following, the parameter settings are described. These values were defined according to a preliminary study.

For the genetic operators, the neighborhood size (T) defines the range of neighbor solutions that can be selected as the parents p_1 and p_2. In accordance to [Chang et al., 2008, Alhindi e Zhang, 2014], we set $T = 10$, probability of crossover $P_c = 1.0$ and probability of the single insert mutation $P_m = 0.5$. Furthermore, for the MEDA/D-MK sampling step (described in Section 4.3), the probability to apply a unique insert-based movement in σ_s^k is also $P_m = 0.5$.

1. *Number of subproblems* (N)*:* A large number of subproblems means a higher computational cost because it increases the number of fitness evaluations at each generation. However, a small number of subproblems can deteriorate the algorithm search ability to explore the different regions of the objective space. Therefore, we set $N = 100$ as it is a reasonable number of weight vectors (subproblems) for solving bi-objective problems.

2. *Update neighborhood:* We have used a scheme that we named *range sensitive global update*, which means that, according to the Euclidean distance between the weight vectors, a new sampled solution σ_s^k tries to update the current population from the closest neighbor to the farther neighbor solution.

3. *Maximum replacements* (n_r)*:* Each sampled solution σ_s^k can update a maximum of $n_r = 2$ subproblems.

4. *Spread parameter (θ):* It is set to assign a probability of 0.8 to the central permutation (σ_0^k) at every generation.

5. *Stopping criterion:* The algorithms stop when a maximum number of evaluations is reached. For each problem, the maximum number of evaluations depends on the problem size as $MaxGen = n \times 1000$.

Moreover, we track the outcome of the algorithms at each $(n \times 1000)/100$ generations by inspecting the current *PF*, i.e., 10 approximated *PFs* are provided and evaluated in each run. In this way, we can evaluate the algorithm's behavior throughout the evolution.

7.4.3 Comparison to the MOEA/D variant

In this section, we have compared MEDA/D-MK to MOEA/D variant, using the scalarizing functions *Weighted sum* and *Tchebycheff*. First, we have provided the average normalized *HV* results obtained by MOEA/D and MEDA/D-MK for each one of the 110 Taillard test instances and ranked their results according to the Kruskall-Wallis statistical test. This table is presented in a in Appendix A. The results show that MEDA/D-MK with *Weighted Sum* achieves the best results with a significant difference in 78 of the 110 test instances. We have summarized these results by grouping the test instances according to the problem scale $n \times m$ (10 instances each). Thus, the analysis of the results present in this section was conducted according to 100 independent runs using the two quality indicators and the non-parametric Friedman statistical test (at 5% significance level) to check if the results obtained have a significant difference. Table 7.2 presents the average *HV* values obtained by MEDA/D-MK and MOEA/D. For each group of instances, the best result, according to the statistical test, is highlighted in boldface. Table 7.3 shows the results of the *C-metric* obtained by the MEDA/D-MK against MOEA/D.

Table 7.2: Average *HV* results

instance	*Weighted Sum*		*Tchebycheff*	
	MOEA/D^w	MEDA/D-MK^w	MOEA/D^t	MEDA/D-MK^t
20x5	0.8087	**0.8728**	0.7804	**0.8413**
20x10	0.8074	**0.8757**	0.7387	0.8289
20x20	0.8021	**0.8577**	0.7151	0.7985
50x5	**0.8678**	**0.9242**	0.7617	0.8381
50x10	0.7449	**0.8378**	0.5818	0.6642
50x20	0.7531	**0.8544**	0.6150	0.6914
100x5	0.8578	**0.8892**	0.7423	0.8151
100x10	0.7550	**0.8161**	0.5764	0.6350
100x20	0.7319	**0.8344**	0.5037	0.5491
200x10	**0.8011**	**0.8258**	0.5885	0.6219
200x20	0.7278	**0.8101**	0.4497	0.4816

The results in Table 7.2 show that *Weighted sum* outperforms *Tchebycheff*. Additionally, MEDA/D-MK with *Weighted sum* (referenced as MEDA/D-MK^w) outperforms MOEA/D^w with a significant difference in 9 of the 11 groups of instances. The groups of test instances for which MEDA/D-MK^w does not outperform MOEA/D^w with a significant difference are 50x5 and 200x10. As Mietthen [Miettinen, 2012] argued, the *Weighted sum* approach is good at convex (concave) bi-objective problems, while *Tchebycheff* approach is useful when the problem is non-convex. This is also confirmed in our results, in which *Weighted sum* outperforms *Tchebycheff*. So, for the remaining experimental studies, we have used the *Weighted sum*.

Table 7.3: *C-metric* values between A = MEDA/D-MK against MOEA/D and the *best-known* reference approximated *PF*. The algorithm that covered more solutions is highlighted in boldface.

instance	B = MOEA/D		B = *reference sets*	
	C(A,B)	C(B,A)	C(A,B)	C(B,A)
20x5	**0.54**	0.03	0.12	**0.15**
20x10	**0.40**	0.06	0.08	**0.12**
20x20	**0.25**	0.07	0.09	**0.11**
50x5	**0.96**	0.02	**0.79**	0.11
50x10	**0.91**	0.07	**0.94**	0.02
50x20	**0.83**	0.12	**0.98**	0.02
100x5	**0.98**	0.02	**0.78**	0.02
100x10	**0.96**	0.01	**0.82**	0.04
100x20	**0.82**	0.15	**1.00**	0.00
200x10	**0.72**	0.16	**0.83**	0.00
200x20	**0.85**	0.13	**1.00**	0.00

In addition, it is evident from Table 7.3 that, according to the *C-metric*, the approximated *PFs* from MEDA/D-MK dominates a large percentage of solutions from the approximated *PFs* obtained by MOEA/D in all the groups of instances.

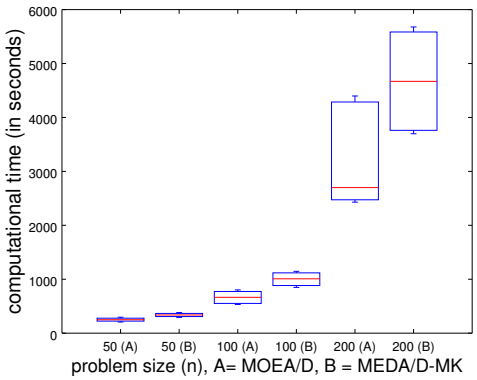


Figure 7.3: Average computational time (in seconds) of MOEA/D (A) and MEDA/D-MK (B) for the merged groups of instances with 50, 100 and 200 jobs, after $nx1000$ generations.

It is explicit from Figure 3 that the process performed by MEDA/D-MK to generate new solutions is more time-consuming compared to the conventional genetic operators. The difference increases as the problem scale (number of jobs) increases because the sampling step from MEDA/D-MK is $O(n^2)$. This behavior is typical for EDAs, which are slower than GAs and other similar approaches. Usually, EDAs are more complex but generally they are able to produce solutions of better quality.

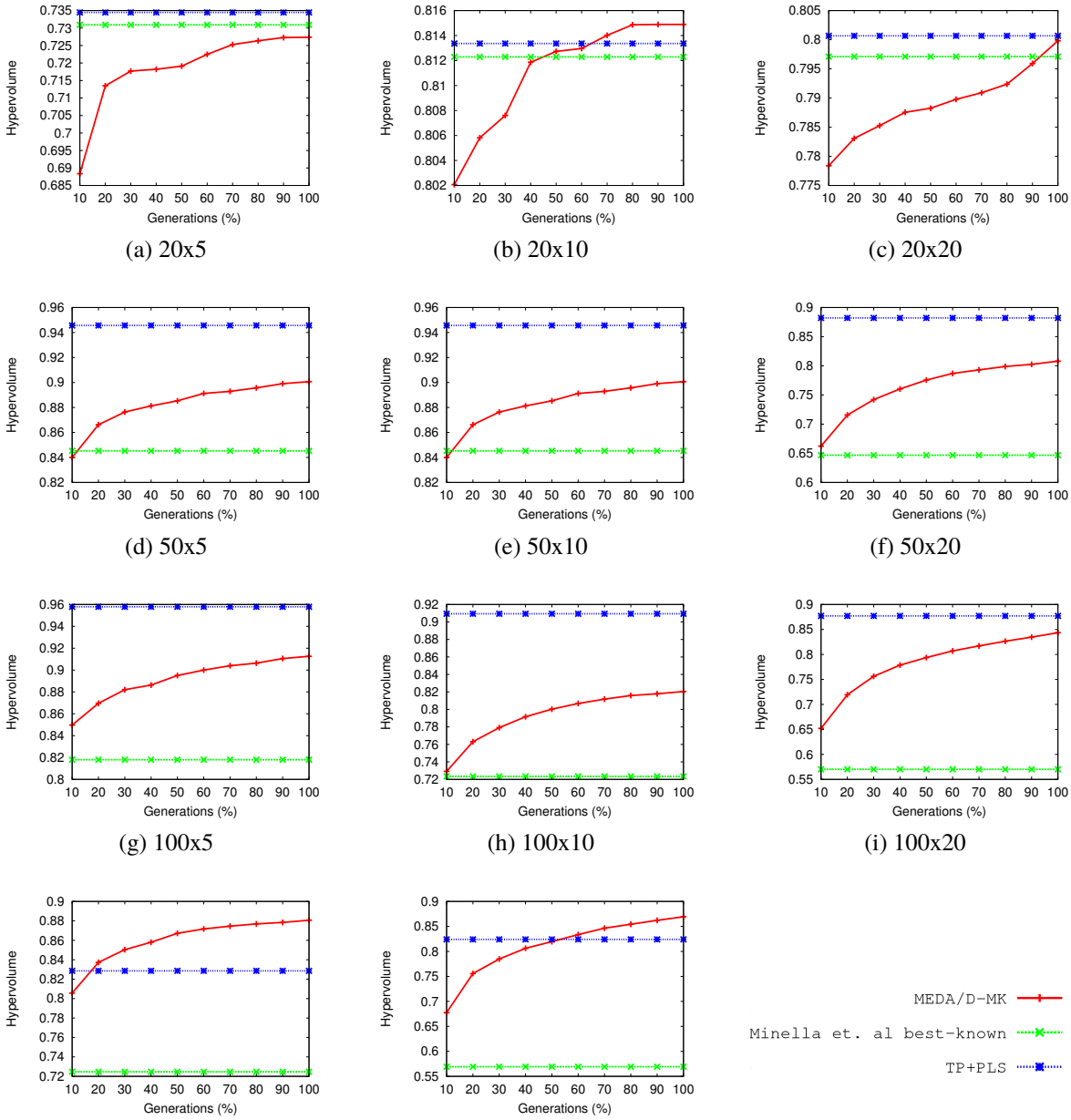
7.4.4 Discussion: Mallows model vs. genetic operators:

In MOEA/D, $B(k)$ defines the range of solutions that can be chosen as parent solutions. It is easy to prove that (at least for MOPs with concave (convex) *PF* shapes) the solutions of two subproblems have a higher level of similarity as their weight vectors are closer. In Mallows models, θ determines the probabilities of sampling a solution close to the central permutation σ_0. Thus, θ and $B(k)$ has a similar effect, in which they can influence the search regarding the sparsity (distance) between the offspring and their parents.

Besides, the sampling from MM and the genetic operators are different regarding the type of variation/movements that they perform: the Cayley distance is based on the minimum number of swaps (not necessarily adjacent) that have to be carried out to transform σ into π, and the genetic operator is based on the two-point crossover.

It is also known that, unlike other EAs, EDAs can provide models expressing the regularities of the problem structure [Larrañaga et al., 2012], being this property one of the primary motivations of using EDAs instead of other EAs.

7.4.5 Comparison to reference sets



(a) 20x5 (b) 20x10 (c) 20x20

(d) 50x5 (e) 50x10 (f) 50x20

(g) 100x5 (h) 100x10 (i) 100x20

(j) 200x10 (k) 200x20 (l) Algorithms

Figure 7.4: Average *HV* values obtained by MEDA/D-MK throughout the generations (every %10) compared to the reference sets from the literature (those from Minella et. al [Minella et al., 2011] and TP+PLS [Dubois-Lacoste et al., 2011]) for the 11 groups of instances (a)-(k)

We devote this section to analyze our approach in the context of the state-of-the-art approaches/results for the benchmark considered. As it has not been possible to obtain the source code of the best-performing approaches (requested to the authors), we have used the *best-known* approximated *PFs* reported by their respective authors. The *best-known* reference sets reported by [Minella et al., 2011][1] were produced by joining all the approximated *PFs* achieved by 7 re-implemented (or adapted) state-of-the-art methods, including their best-performing proposed algorithm RIPG, different parameters configurations, and several runs. [Dubois-Lacoste et al., 2011][2] provided the reference sets for each algorithm evaluated, including their best-performing algorithm TP+PLS. In order to have comparable sets, we have followed the same procedure, composing our reference sets by merging the outcomes from MEDA/D-MK.

It must be noted that the results shown in this section must be interpreted carefully, since the reference sets were obtained with different stopping conditions and merging different runs/algorithms. Therefore, we have compared MEDA/D-MK to the reference sets throughout the generations. This comparison allow us to observe the dynamics of the MEDA/D-MK during the search.

Figure 7.4 shows the average *HV* values obtained by MEDA/D-MK throughout the generations (10 plots) compared to the average *HV* values obtained from the reference sets (constant lines) for the group of instances. According to the results, we can make the following remarks:

- In general, MEDA/D-MK is able to keep evolving throughout all the generations. However, the algorithm slowly improves the results after 70% of the generations (except for the group of instances 20×20). If source codes of the algorithms were available, it would be interesting to analyze the ability of each algorithm to evolve throughout generations.

- The reference sets from Dubois-Lacoste et al. outperform the reference sets from Minella et al. in all the cases. The difference increases as the problem scale increases.

- MEDA/D-MK achieves competitive results. Our approach easily outperforms the reference sets from Minella et al. in all the cases, except for the 20×5. Also, it outperforms the reference sets from Dubois-Lacoste et al. for many of the instances of size 20, and for almost all the instances of size (e.g., 200 jobs). The HV values for each particular instance are provided in the supplementary material.

- This type of probabilistic models, like the Mallows model we introduced, have a higher complexity than those of its competitors. However, as it has been demonstrated in other works on EDAs [Ceberio et al., 2014a], even if they are usually slower than other approaches, they are able to *continue evolving* and thus obtain better results.

To complement the information presented in Figure 7.4, Table 7.4 presents a comparison (by pairs) based on the final *HVs* obtained by the three different approaches for each of one the 110 test instances. In this table, for each group of instances, we count how many times one approach outperforms the other. Regarding that, each group contains 10 instances. Looking at the results, it can be seen that MEDA/D-MK and TP+PLS systematically outperform the results provided by Minella's reference sets. Regarding MEDA/D-MK and TP-PLS, the later shows better results for the groups with 50 and 100 jobs, while the former behaves better for $20 \times 10, 200 \times 10$ and 200×20.

[1] Minella et al.: http://soa.iti.es.

[2] Dubois-Lacoste et al.: http://iridia.ulb.ac.be/~jdubois/pfsp_refsets.htm

Finally, Figure 7.5 presents the plots of the final approximated *PFs* obtained by the algorithms for four representative large-scale test instances (Ta071, Ta082, Ta091 and Ta101). From this figure, we can observe that the *PFs* are concave, and that the cardinality of the fronts increases with the number of machines. As expected, the best approximated *PFs* are those obtained by MEDA/D-MK and TP+PLS. In general, TP+PLS find more solutions that covers the extremes regions of the PFs because of its search behavior, in which the first phase of the method applies a single-solution algorithm to find high-quality solutions for each objective function separately.

Table 7.4: Comparison between pairs of algorithms according to the *HV* measure. For each pair of algorithms × group of instances, a cell indicates three values: 1) the number of times that the final *HV* of approach *A* is better than that of approach *B*, 2) the number of draws, 3) the number of times that the *HV* of approach *B* is better than that of approach *A*). Reference sets: ***A* = MEDA/D-MK, *B* = Minella et. al best-known, *C* = Dubois-Lacoste et al. (TP+PLS)**

Instances	*A-draw-B*	*A-draw-C*	*C-draw-B*
20x5	2-5-3	1-3-6	6-2-2
20x10	6-3-1	6-3-1	4-2-4
20x20	6-1-3	5-3-2	6-1-3
50x5	9-0-1	0-0-10	10-0-0
50x10	10-0-0	0-0-10	10-0-0
50x20	10-0-0	0-0-10	10-0-0
100x5	9-0-1	3-1-6	10-0-0
100x10	10-0-0	1-0-9	10-0-0
100x20	10-0-0	2-0-8	10-0-0
200x10	10-0-0	9-0-1	10-0-0
200x20	10-0-0	8-0-2	10-0-0
Total	**92-9-9**	**35-10-65**	**96-5-9**

To ease the comparison to other methods, we have compiled our *best-known* reference approximated *PFs*. Each reference set was produced by merging all the MEDA/D-MK results regarding all runs and all parameter configurations. Our 110 *best known* reference sets are available on-line[3].

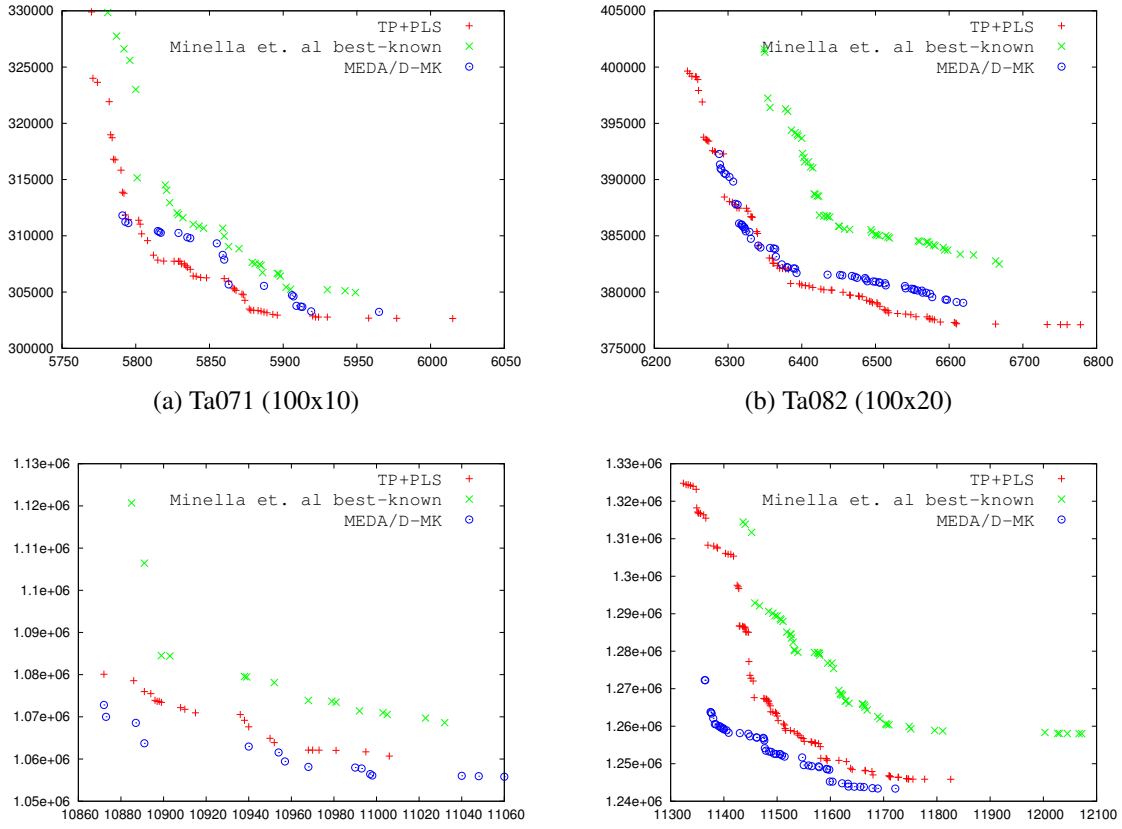
7.4.6 Influence of the MEDA/D-MK components

In this section, in order to further illustrate the dynamics of MEDA/D-MK and provide an understanding of the parameters used, we present the study of (i) the spread parameter θ, (ii) the constructive algorithm $LR(n/m)$ to initialize the population, and (iii) the controlled shaking procedure used to escape from the local optima.

The spread parameter is directly related to the shape of the exponential probabilistic model [Irurozki et al., 2014a]. Thus, we have evaluated different θ values in order to assign the following probabilities to the central permutation $P(\sigma_0) = \{0.5, 0.6, 0.7, 0.8\}$ that depends on the number of variables (see Eq.(7.4)).

Table 7.5 presents the θ values for the different problem sizes and their corresponding probabilities of sampling $P(\sigma_0)$. Table 7.6 shows that, regarding the *HV* indicator, MEDA/D-MK with a higher $P(\sigma_0)$ achieves the best results. Besides, for $P(\sigma_0) = 0.7$ and $P(\sigma_0) = 0.8$ the

[3]Available at https://github.com/MuriloZangari/supplementary_results_mopfsp



(a) Ta071 (100x10)

(b) Ta082 (100x20)

(c) Ta091 (200x10)

(d) Ta101 (200x20)

Figure 7.5: Illustrative plots of the final approximated *PFs* by MEDA/D-MK, Minella et. al best-known and TP+PLS for (a) Ta071, (b) Ta081, (c) Ta091, and (d) Ta101 instances.

Table 7.5: The θ values using the Cayley distance for different problem sizes (n) and the respective probabilities $P(\sigma_0)$ assigned.

n	$P(\sigma_0) = 0.5$	$P(\sigma_0) = 0.6$	$P(\sigma_0) = 0.7$	$P(\sigma_0) = 0.8$
20	5.60	5.90	6.30	6.80
50	7.48	7.78	8.20	8.60
100	8.90	9.20	9.60	10.01
200	10.3	10.60	11.00	11.41

difference is not significant in 10 of the 11 cases. It is worth noting that we have used the strategy that avoids the central permutation to be sampled to prevent many copies in the population. A higher $P(\sigma_0)$ means that the probability of sampling a solution close to the σ_0 is higher (low sparsity between permutation solutions), which means that the algorithm conducts an exploitation in each kernel. The other components/ingredients of the framework may promote the exploration in different areas of the search space.

Table 7.7 shows that both MEDA/D-MK and MOEA/D achieve better results using the $LR(n/m)$ initialization (referenced as MOEA/D$_{LR}$ and MEDA/D-MK$_{LR}$), mainly for the larger test instances. Overall, the MEDA/D-MK$_{LR}$ produces the best results. For the smallest instances $n = \{20, 50\}$, MEDA/D-MK does not achieve a significant difference with and without the constructive initialization. There are two possible explanations for this behavior: the $LR(n/m)$ initialization can quickly approximate the solutions to the true *PF*, or the exploration-like behavior

Table 7.6: HV values of the analysis of the MEDA/D-MK regarding the θ parameter.

instance	$P(\sigma_0) = 0.5$	$P(\sigma_0) = 0.6$	$P(\sigma_0) = 0.7$	$P(\sigma_0) = 0.8$
20x5	**0.7536**	**0.7661**	**0.7584**	**0.7575**
20x10	**0.8255**	**0.8261**	**0.8300**	**0.8297**
20x20	**0.7907**	**0.7910**	**0.7969**	**0.7958**
50x5	**0.9057**	**0.9048**	**0.8998**	**0.9008**
50x10	0.7785	0.7793	**0.8026**	**0.7879**
50x20	0.7737	0.7713	**0.8065**	**0.8074**
100x5	**0.8203**	**0.8191**	**0.8308**	**0.8007**
100x10	**0.7786**	**0.7907**	**0.7955**	**0.7989**
100x20	0.7006	**0.7599**	**0.7645**	**0.7655**
200x10	0.7678	**0.7822**	**0.7923**	**0.7972**
200x20	0.6543	0.6531	0.6782	**0.6883**

Table 7.7: HV results of the analysis using the constructive algorithm $LR(n/m)$ to initialize the population.

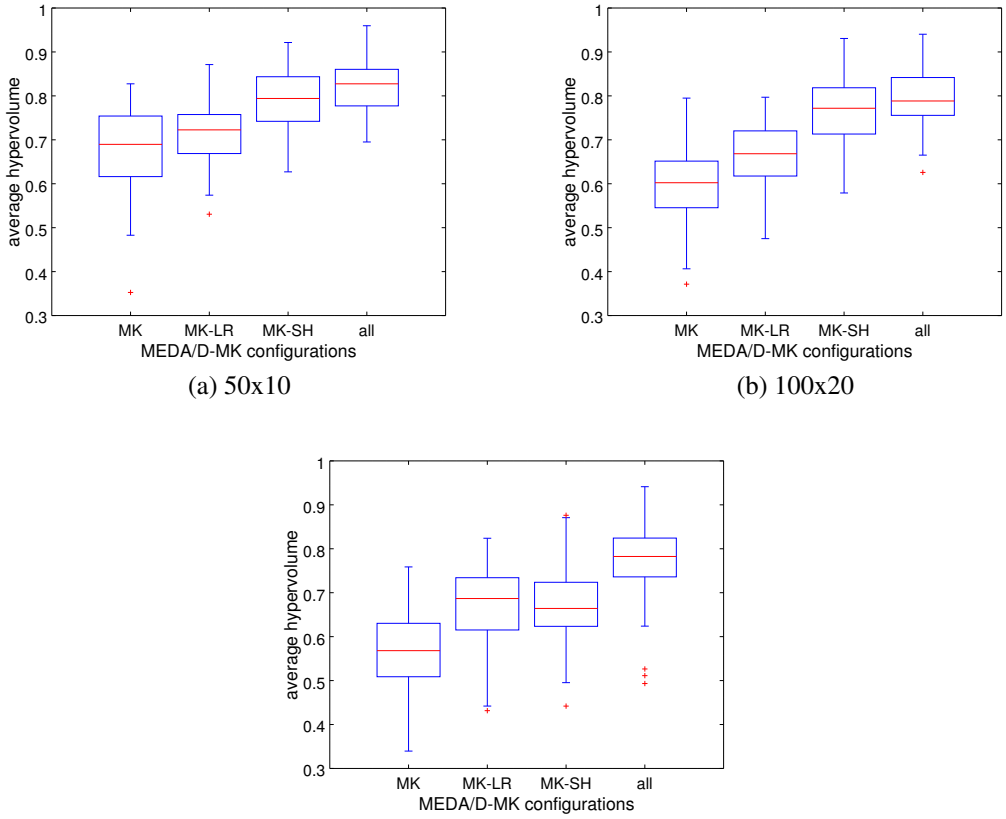
instance	MOEA/D	MOEA/D$_{LR}$	MEDA/D-MK	MEDA/D-MK$_{LR}$
20x5	**0.7342**	**0.7359**	**0.7509**	**0.7590**
20x10	**0.8372**	**0.8346**	**0.8467**	**0.8512**
20x20	0.8167	0.8086	**0.8281**	**0.8245**
50x5	0.8712	0.8862	**0.9079**	**0.9165**
50x10	0.6936	0.7369	**0.7735**	**0.7984**
50x20	0.6605	0.7121	**0.7751**	**0.8005**
100x5	0.7143	0.8444	0.8184	**0.8804**
100x10	0.6587	0.7751	0.7654	**0.8560**
100x20	0.5881	0.6528	**0.7469**	**0.7677**
200x10	0.5251	**0.8064**	0.6204	**0.8452**
200x20	0.5460	0.6852	0.6481	**0.7610**

Table 7.8: HV results of analysis using the controlled shake procedure.

instance	MOEA/D	MOEA/D$_{SH}$	MEDA/D-MK	MEDA/D-MK$_{SH}$
20x5	**0.7503**	**0.7499**	**0.7658**	**0.7725**
20x10	**0.8339**	**0.8335**	**0.8351**	**0.8501**
20x20	0.8053	0.8050	0.8039	**0.8201**
50x5	0.8883	0.8869	0.9026	**0.9245**
50x10	0.7240	0.7420	0.7433	**0.8061**
50x20	0.6866	0.7205	0.7109	**0.8140**
100x5	0.7956	0.7932	0.8298	**0.8577**
100x10	0.7266	0.7458	0.7707	**0.8422**
100x20	0.6003	0.6454	0.6502	**0.7636**
200x10	0.7388	0.7420	0.7635	**0.7933**
200x20	0.5796	0.6396	0.6436	**0.7196**

of the algorithms is not efficient enough to evolve the population for the smallest problem scales. We should further investigate this aspect.

The shaking procedure has two additional parameters to be set. To avoid the complexity of the parameters setting, we set them according to the number of jobs (n). After n generations without σ^k being improved, the procedure executes the perturbation based on $(n/10)$ insert-based movements.



(a) 50x10

(b) 100x20

(c) 200x20

Figure 7.6: Boxplot of the average HV obtained by 4 MEDA/D-MK configurations: (i) without both the components (MK), (ii) MEDA/D-MK with $LR(n/m)$ (MK-LR), (iii) MEDA/D-MK with shaking procedure (MK-SH), and (iv) MEDA/D-MK with the both components (all).

Table 7.8 shows that MEDA/D-MK with the shaking procedure (MEDA/D-MK$_{SH}$) produces the best HV results with a significant difference in all the cases, except for 20x5, and 20x10. Overall, the results show that the search needs an efficient balance between the exploitation and exploration to keep converging to the true PF while it is able to escape from local optima.

Figure 7.6 complements these results, where we compare four different MEDA/D-MK configurations regarding its components to initialize the population and the shaking procedure. The results indicate that the shaking procedure has a higher positive influence compared to the constructive $LR(n/m)$ initialization. Besides, the both components applied together really improved the algorithm.

7.5 Conclusion and Future Work

Recently, EDAs that incorporate distance-based exponential probability models over permutations have been proposed for solving different single-objective permutation optimization problems efficiently. In this paper, we have presented, for the first time, a general multi-objective estimation of distribution algorithm based on decomposition and Kernels of Mallows models (MEDA/D-MK framework).

In order to demonstrate the viability of the proposal to solve multi-objective permutation-based problems, we have applied it on 110 MoPFSP test instances minimizing makespan and total flow time. The MEDA/D-MK for MoPFSP is enhanced with a constructive PFSP-specific procedure to initialize the solutions, and a perturbation procedure to control the exploration/exploitation for finding a diverse set of non-dominated solutions.

The experimental study shows that MEDA/D-MK outperforms the tailored MOEA/D for MoPFSP. Thus, we have showed the potentiality of using Mallows Models EDA in the context of the MOEA/D framework. Moreover, the analysis demonstrated that our approach achieved competitive results compared to the reference sets reported in the literature for the benchmark considered. For the large-scale test instances, MEDA/D-MK has significantly produced better approximated *PFs* using a reasonable number of evaluations and computational cost. Furthermore, for reproducibility purposes, we have produced our *best-known* results, being they available at https://github.com/MuriloZangari/supplementary_results_mopfsp for future comparison to other approaches.

Departing from the significant results reported in this paper, we state some trends for future work: (i) the analysis of other permutation distance metrics such as the Ulam distance [Irurozki et al., 2014b], (ii) investigate the efficiency of our approach to deal with the MoPFSP involving more objectives (e.g., the total tardiness), and (iii) to solve other permutation problems, e.g., the multi-objective Quadratic Assign Problem.

In addition, components of our algorithm, such as the Mallows models, could also be incorporated to recent methods proposed for automatically designing MOEAs. In particular, they could be included in the AutoMOEA template [Bezerra et al., 2016], a recent component-wise design of multi-objective algorithms, which is able to combine automatically different procedures and components, creating novel MOEA algorithms.

Chapter 8

Final considerations

In this thesis, we have proposed hybrid metaheuristics for solving MCOPs. Our frameworks are considered MOEA/D variants because they follow the main guidelines from MOEA/D, e.g., the decomposition of the problem at hand into a number of scalar single-objective subproblems, and the use of the neighborhood concept to evolve the subproblems in a collaborative manner. We have instantiated different models (metaheuristics) as the *variation step* in MOEA/D, and we evaluated the potentiality of using these models instead of *crossover* and *mutation*. Further, for each kind of problem addressed, we have incorporated a number of design modifications into the framework aiming to improve their search ability, and thus achieving better results than the state-of-the-art.

Depending on the problem at hand, one model can be more suitable than the others. We have distinguished our proposal into three frameworks: (i) the MOEA/D-BACO, which we have applied solve the mUBQP, (ii) the MOEA/D-GM, which is able to instantiate different probabilistic models, including graphical models, for solving complex (e.g., deceptive) problems with binary representation where, usually, traditional genetic operators fail to efficiently solve them, and (iii) the MEDA/D-MK which is proposed for solving multi-objective permutation problems.

The MOEA/D-BACO framework was firstly proposed to solve any MCOP with binary representation. We have evaluated it on several mUBQP large-scale test instances. We have also included the components (i) mutation-like effect, (ii) a diversity preserving mechanism, and (iii) the varied neighborhood size to improve the algorithm search ability. The experimental results show that MOEA/D-BACO significantly outperforms the MOEA/D in most of the test instances. Moreover, we have evaluated the impact of using the mUBQP specific local search within MOEA/D-BACO. We have also compared our proposal to a set of *best-known* results for the benchmark considered. The results show the potentiality of our approach to solve the problem which is able to balance the *exploration vs. exploitation*.

The MOEA/D-GM framework has been proposed as a general MOEA/D extension able to instantiate well-known probabilistic models from EDAs. In the MOEA/D-GM, we can instantiate both univariate and multivariate probabilistic models for each scalar optimization subproblem. To validate the introduced framework, an experimental study was conducted on a multi-objective version of the deceptive function *Trap5*. The results indicate that the MOEA/D-Tree instantiation, in which tree models are learned from the matrices of mutual information between the variables, is able to capture the structure of the problem, and thus outperforming the MOEA/D with univariate models. However, for solving the mUBQP, the framework using the univariate models, such as PBIL, outperforms the MOEA/D-Tree.

Finally, the MEDA/D-MK is a novel general framework proposed to solve any multi-objective permutation-based optimization problem. In order to demonstrate the validity of the MEDA/D-MK, we applied it to solve the MoPFSP. The results show that MEDA/D-MK outperforms the MOEA/D variant specially tailored for the MoPFSP. Furthermore, our approach is able to outperform the *best-known* approximated PFs reported in the literature for the benchmark considered.

Overall, we show that the MOEA/D guidelines hybridized to other metaheuristics (e.g., that incorporates probabilistic models) is a powerful strategy for solving MCOPs.

We have distinguished our contributions in different papers. The complete list of paper is as follows:

Publications in Conferences:

1. [Zangari e Pozo, 2014a] Parallel MOEA/D-ACO on GPU. In *Advances in Artificial Intelligence (IBERAMIA)*, pages 405–417. Springer (2014).

2. [Zangari e Pozo, 2014b] A GPU Implementation of MOEA/D-ACO for the multi-objective Traveling Salesman Problem. In *Brazilian Conference on Intelligent Systems (BRACIS)* pages 324–329. IEEE (2014).

3. [Zangari e Pozo, 2015] Multi-objective Binary ACO for Unconstrained Binary Quadratic Programming. In *Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS)*, pages 86–91. IEEE (2015).

4. [Zangari et al., 2015a] PBIL: un mismo nombre para distintos algoritmos. Un caso de estudio sobre un problema de optimización multi-objetivo. In *Proceedings of the Spanish Asociation in Artificial Inteligence (CAEPIA)*, pages 283-295, Albacete, Spain (2015).

5. [Zangari et al., 2016b] On the design of hard mUBQP instances. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 421–428, New York, NY, USA. ACM (2016)

Publication is Journals:

1. [Zangari et al., 2017] Not all PBILs are the same: Unveiling the different learning mechanism of PBIL variants. Submitted for publication. In Applied Soft Computing, pages 88-96, volume 53, (2017).

Accepted for publication in journals:

1. [Zangari et al., 2016a] A Decomposition-based Binary ACO algorithm for the multi-objective UBQP. Accept for publication in *Neurocomputing*. (2016)

Technical Report:

1. [Zangari et al., 2015b] MOEA/D-GM: Using Probabilistic Graphical Models in MOEA/D for solving Combinatorial Optimization Problems. Technical Report arXiv:1511.05625, Department of Computer Science and Artificial Intelligence, University of the Basque Country (2015)

Under revision:

1. Multi-objective Decomposition-based Mallows Models Estimation of Distribution Algorithm. A case of study for Permutation Flowshop Scheduling Problem. Submitted for publication. (2016).

8.1 Limitations

As might be expected, we have faced some limitations during our research. For instance, the shortage of source codes of the state-of-the-art algorithms. The availability of the source codes is fundamental to investigate the efficiency of a new approach. We have eliminated the possibility of implementing various algorithms due to our time constraint. In fact, one of the major shortcomings in our area of research is the efficient integration of algorithms, problems, and results to facilitate further investigations. Besides, we make our best results available for future research.

As Blum et al. discussed in [Blum et al., 2011], one of the main arguments in favor of metaheuristics has always been their generality. However, over time, the focus of several researches in metaheuristics and their applications has shifted towards their performance, at the cost of loosing generality. The generalization of metaheuristics to solve single-objective, multi-objective and many-objective optimization problems on discrete and continuous domain is a difficult task, and the efforts to overcome this issue is an important research direction.

8.2 Future work

As future work, we have numerous possibilities. The main future work is to investigate our MOEA/D variants to solve other problems. We can analyze MOEA/D-BACO and MOEA/D-GM to solve other combinatorial problems with binary representation, and MEDA/D-MK to solve other problems based permutation, such as the quadratic assignment problem.

Furthermore, the hyper-heuristics concepts can be used to improve the generalization of the approaches, so that the framework can dynamically (automatically) set different design choices of the MOEA/D, such as the neighborhood size for selection and update, the maximum number of replacements by a new solution in the update step, to set the most appropriated scalarizing function for a given problem, as well as the application of the different models (in this case, each subproblem could apply diverse models throughout the generations intending to find the best one).

Another relevant aspect is the use of more complex models in detriment of the computational cost. Therefore, parallel designs could be a subject of matter. Moreover, we intend to make available our source codes to the research community.

Bibliography

[Aguirre e Tanaka, 2007] Aguirre, H. E. e Tanaka, K. (2007). Working principles, behavior, and performance of MOEAs on MNK-landscapes. *European Journal of Operational Research*, 181(3):1670–1690.

[Alhindi e Zhang, 2014] Alhindi, A. e Zhang, Q. (2014). MOEA/D with Tabu search for multiobjective permutation flow shop scheduling problems. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1155–1164. IEEE.

[Arroyo e Armentano, 2005] Arroyo, J. E. C. e Armentano, V. A. (2005). Genetic local search for multi-objective flowshop scheduling problems. *European Journal of Operational Research*, 167(3):717–738.

[Arroyo e Souza Pereira, 2011] Arroyo, J. E. C. e Souza Pereira, A. A. (2011). A grasp heuristic for the multi-objective permutation flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 55(5-8):741–753.

[Baluja, 1994] Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Relatório Técnico CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA.

[Baluja, 1997] Baluja, S. (1997). Genetic algorithms and explicit search statistics. *Advances in Neural Information Processing Systems*, 9:319–325.

[Baluja e Caruana, 1995] Baluja, S. e Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In *Machine Learning: Proceedings of the Twelfth International Conference*, pages 38–46.

[Baluja e Davies, 1997] Baluja, S. e Davies, S. (1997). Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In Fisher, D. H., editor, *Proceedings of the 14th International Conference on Machine Learning*, pages 30–38.

[Battiti et al., 2008] Battiti, R., Brunato, M. e Mascia, F. (2008). *Reactive search and intelligent optimization*, volume 45. Springer Science & Business Media.

[Beume et al., 2007] Beume, N., Naujoks, B. e Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669.

[Bezerra et al., 2016] Bezerra, L. C. T., nez, M. L.-I. e Stützle, T. (2016). Automatic component-wise design of multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(3):403–417.

[Blum et al., 2011] Blum, C., Puchinger, J., Raidl, G. R. e Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151.

[Blum e Roli, 2003] Blum, C. e Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308.

[Blum e Roli, 2008] Blum, C. e Roli, A. (2008). Hybrid metaheuristics: an introduction. In *Hybrid Metaheuristics*, pages 1–30. Springer.

[Botev et al., 2013] Botev, Z., Kroese, D. P., Rubinstein, R. Y., L'Ecuyer, P. et al. (2013). *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation*. North-Holland Mathematical Library-Elsevier.

[Brooks e Morgan, 1995] Brooks, S. P. e Morgan, B. J. (1995). Optimization using simulated annealing. *The Statistician*, pages 241–257.

[Brownlee et al., 2012] Brownlee, A. E. I., McCall, J. e Pelikan, M. (2012). Influence of selection on structure learning in Markov network EDAs: An empirical study. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 249–256. ACM.

[Burke et al., 2013] Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. e Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.

[Ceberio et al., 2012] Ceberio, J., Irurozki, E., Mendiburu, A. e Lozano, J. A. (2012). A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. *Progress in Artificial Intelligence*, 1(1):103–117.

[Ceberio et al., 2014a] Ceberio, J., Irurozki, E., Mendiburu, A. e Lozano, J. A. (2014a). A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. *IEEE Transactions on Evolutionary Computation*, 18(2):286–300.

[Ceberio et al., 2014b] Ceberio, J., Irurozki, E., Mendiburu, A. e Lozano, J. A. (2014b). Extending distance-based ranking models in estimation of distribution algorithms. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2459–2466. IEEE.

[Ceberio et al., 2011] Ceberio, J., Mendiburu, A. e Lozano, J. A. (2011). Introducing the Mallows model on estimation of distribution algorithms. In *Neural Information Processing*, pages 461–470. Springer.

[Ceberio et al., 2013] Ceberio, J., Mendiburu, A. e Lozano, J. A. (2013). The Plackett-Luce ranking model on permutation-based optimization problems. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 494–501. IEEE, IEEE press.

[Ceberio et al., 2015a] Ceberio, J., Mendiburu, A. e Lozano, J. A. (2015a). Kernels of Mallows Models for solving permutation-based problems. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 505–512. ACM.

[Ceberio et al., 2015b] Ceberio, J., Santana, R., Mendiburu, A. e Lozano, J. A. (2015b). Mixtures of Generalized Mallows models for solving the quadratic assignment problem. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2050–2057. IEEE.

[Cecilia et al., 2013] Cecilia, J. M., Garcia, J. M. e Nisbet (2013). Enhancing data parallelism for ant colony optimization on gpus. *J. Parallel Distrib. Comput.*, 73(1):42–51.

[Chang et al., 2008] Chang, P. C., Chen, S. H., Zhang, Q. e Lin, J. L. (2008). MOEA/D for flowshop scheduling problems. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 1433–1438. IEEE.

[Chaves-González et al., 2008] Chaves-González, J. M., Vega-Rodríguez, M. A., Domínguez-González, D., Gómez-Pulido, J. A. e Sánchez-Pérez, J. M. (2008). SS vs PBIL to solve a real-world frequency assignment problem in GSM networks. In *Applications of Evolutionary Computing*, pages 21–30. Springer.

[Chen et al., 2016] Chen, M.-H., Chang, P.-C. e Wu, J.-L. (2016). A population-based incremental learning approach with artificial immune system for network intrusion detection. *Engineering Applications of Artificial Intelligence*.

[Coello et al., 2007] Coello, C., Lamont, G. e Van Veldhuizen, D. (2007). *Evolutionary Algorithms for Solving Multi-objective Problems*. Springer-Verlag New York Inc.

[Crainic e Toulouse, 2010] Crainic, T. G. e Toulouse, M. (2010). Parallel meta-heuristics. In *Handbook of metaheuristics*, pages 497–541. Springer.

[da Silva e Schirru, 2014] da Silva, M. H. e Schirru, R. (2014). A self-adaptive quantum PBIL method for the nuclear reload optimization. *Progress in Nuclear Energy*, 74:103–109.

[Dawson e Stewart, 2013] Dawson, L. e Stewart, I. A. (2013). Improving ant colony optimization performance on the gpu using cuda. In *IEEE Congress on Evol. Comp.*, pages 1901–1908.

[Deb e Goldberg, 1993] Deb, K. e Goldberg, D. E. (1993). Analyzing deception in trap functions. *Foundations of genetic algorithms*, 2:93–108.

[Deb e Jain, 2014] Deb, K. e Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Transaction Evolutionary Computation*, 18(4):577–601.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S. e Meyarivan, T. (2002). A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197.

[Deb et al., 2005] Deb, K., Thiele, L., Laumanns, M. e Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pages 105–145. Springer London.

[DeléVacq et al., 2013] DeléVacq, A., Delisle, P., Gravel, M. e Krajecki, M. (2013). Parallel ant colony optimization on graphics processing units. *Journal of Parallel and Distributed Computing*, 73(1):52–61.

[Delevacq et al., 2013] Delevacq, A., Delisle, P., Gravel, M. e Krajecki, M. (2013). Parallel ant colony optimization on graphics processing units. *J. Parallel Distrib. Comput.*, 73(1):52–61.

[Derrac et al., 2011] Derrac, J., García, S., Molina, D. e Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

[Dorigo et al., 2008] Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T. e Winfield, A. (2008). *Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings*, volume 5217. Springer.

[Dorigo et al., 2006] Dorigo, M., Birattari, M. e Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39.

[Dorigo e Gambardella, 1997] Dorigo, M. e Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66.

[Dorigo et al., 1996] Dorigo, M., Maniezzo, V. e Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41.

[dos Santos Coelho e Grebogi, 2010] dos Santos Coelho, L. e Grebogi, R. B. (2010). Chaotic synchronization using PID control combined with population based incremental learning algorithm. *Expert Systems with Applications*, 37(7):5347–5352.

[Dubois-Lacoste et al., 2011] Dubois-Lacoste, J., López-Ibáñez, M. e Stützle, T. (2011). A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. *Computers & Operations Research*, 38(8):1219–1236.

[Echegoyen et al., 2007] Echegoyen, C., Lozano, J. A., Santana, R. e Larrañaga, P. (2007). Exact bayesian network learning in estimation of distribution algorithms. In *2007 IEEE Congress on Evolutionary Computation*, pages 1051–1058. IEEE.

[Ehrgott, 2006] Ehrgott, M. (2006). *Multicriteria optimization*. Springer Science & Business Media.

[Ehrgott e Gandibleux, 2008] Ehrgott, M. e Gandibleux, X. (2008). Hybrid metaheuristics for multi-objective combinatorial optimization. In *Hybrid metaheuristics*, pages 221–259. Springer.

[Feo e Resende, 1995] Feo, T. A. e Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.

[Fernandes et al., 2007] Fernandes, C. M., Rosa, A. C. e Ramos, V. (2007). Binary ant algorithm. In *Proceedings of the 9th Conference on Genetic and Evolutionary Computation (GECCO)*, pages 41–48. ACM.

[Fligner e Verducci, 1988] Fligner, M. A. e Verducci, J. S. (1988). Multistage ranking models. *Journal of the American Statistical association*, 83(403):892–901.

[Folly, 2013] Folly, K. A. (2013). Parallel PBIL applied to power system controller design. *Journal of Artificial Intelligence and Soft Computing Research*, 3(3):215–223.

[Folly, 2014] Folly, K. A. (2014). Comparison of multi-population PBIL and adaptive learning rate PBIL in designing power system controller. In *Advances in Swarm Intelligence*, pages 135–145. Springer.

[Fritsche et al., 2015] Fritsche, G., Strickler, A., Pozo, A. e Santana, R. (2015). Capturing relationships in multi-objective optimization. In *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, pages 222–227. IEEE.

[Gandibleux e Freville, 2000] Gandibleux, X. e Freville, A. (2000). Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: the two objectives case. *Journal of Heuristics*, 6(3):361–383.

[Giagkiozis et al., 2014] Giagkiozis, I., Purshouse, R. C. e Fleming, P. J. (2014). Generalized decomposition and cross entropy methods for many-objective optimization. *Inf. Sci.*, 282:363–387.

[Glover, 1986] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549.

[Glover, 1989] Glover, F. (1989). Tabu search-part i. *ORSA Journal on computing*, 1(3):190–206.

[Glover e Hao, 2010] Glover, F. e Hao, J.-K. (2010). Efficient evaluations for solving large 0-1 unconstrained quadratic optimisation problems. *International Journal of Metaheuristics*, 1(1):3–10.

[Glover et al., 1998] Glover, F., Kochenberger, G. A. e Alidaee, B. (1998). Adaptive memory tabu search for binary quadratic programs. *Management Science*, 44(3):336–345.

[Glover e Kochenberger, 2006] Glover, F. W. e Kochenberger, G. A. (2006). *Handbook of metaheuristics*, volume 57. Springer Science & Business Media.

[González et al., 2001a] González, C., Lozano, J. A. e Larrañaga, P. (2001a). Analyzing the population based incremental learning algorithm by means of discrete dynamical systems. *Complex Systems*, 12(4):465–479.

[González et al., 2001b] González, C., Lozano, J. A. e Larrañaga, P. (2001b). The convergence behavior of the PBIL algorithm: a preliminary approach. In *Artificial Neural Nets and Genetic Algorithms*, pages 228–231. Springer.

[Hansen, 1997] Hansen, M. P. (1997). Tabu search for multiobjective optimization: Mots. In *Proceedings of the 13th International Conference on Multiple Criteria Decision Making*, pages 574–586. Citeseer.

[Hansen e Ostermeier, 1996] Hansen, N. e Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 312–317.

[Hansen e Mladenović, 2001] Hansen, P. e Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467.

[Hansen e Mladenović, 2003] Hansen, P. e Mladenović, N. (2003). Variable neighborhood search. In Glover, F. e Kochenberger, G., editores, *Handbook of Metaheuristics*, pages 145–184. Kluwer Academic Publisher.

[Hertz, 1991] Hertz, J. (1991). *Introduction to the theory of neural computation*, volume 1. Basic Books.

[Hoffman et al., 2013] Hoffman, K. L., Padberg, M. e Rinaldi, G. (2013). Traveling salesman problem. In *Encyclopedia of Operations Research and Management Science*, pages 1573–1578. Springer.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI.

[Hu et al., 2011] Hu, G., Xiong, W.-Q., Zhang, X. e Yuan, J.-L. (2011). Binary ant colony algorithm with controllable search bias. *Control Theory & Applications*, 28(8):1071–1080.

[Huband et al., 2006] Huband, S., Hingston, P., Barone, L. e While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *Evolutionary Computation, IEEE Transactions on*, 10(5):477–506.

[Irurozki et al., 2014a] Irurozki, E., Calvo Molinos, B. e Lozano Alonso, J. A. (2014a). Sampling and learning the Mallows and Generalized Mallows models under the Cayley distance. Relatório técnico, Department of Computer Science and Artificial Intelligence, University of the Basque Country, San Sebastian, Spain.

[Irurozki et al., 2014b] Irurozki, E., Calvo Molinos, B. e Lozano Alonso, J. A. (2014b). Sampling and learning the Mallows and Generalized Mallows models under the Ulam distance. Relatório técnico, Department of Computer Science and Artificial Intelligence, University of the Basque Country, San Sebastian, Spain.

[Ishibuchi et al., 2013] Ishibuchi, H., Akedo, N. e Nojima, Y. (2013). A Study on the Specification of a Scalarizing Function in MOEA/D for Many-Objective Knapsack Problems. In *Proceedings of the 7th International Learning and Intelligent Optimization Conference (LION)*, volume 7997 de *Lecture Notes in Computer Science*, pages 231–246. Springer.

[Ishibuchi et al., 2016] Ishibuchi, H., Setoguchi, Y., Masuda, H. e Nojima, Y. (2016). Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes. *IEEE Transactions on Evolutionary Computation*. Accpeted for publication.

[Ishibuchi et al., 2003] Ishibuchi, H., Yoshida, T. e Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223.

[Ishubuchi e Murata, 1998] Ishubuchi, H. e Murata, T. (1998). Multi-objective Genetic Local Search Algorithm and its Application to Flowshop scheduling. *IEEE Transaction System Man and Cybernetics*, 28:392–409.

[Johnson, 1985] Johnson, D. S. (1985). The np-completeness column: an ongoing guide. *Journal of Algorithms*, 6(3):434–451.

[Kang et al., 2014] Kang, B.-Y., Xu, M., Lee, J. e Kim, D.-W. (2014). ROBIL: Robot path planning based on PBIL algorithm. *International Journal of Advanced Robotic Systems*, 11:147.

[Karshenas et al., 2014] Karshenas, H., Santana, R., Bielza, C. e Larrañaga, P. (2014). Multi-objective optimization based on joint probabilistic modeling of objectives and variables. *IEEE Transactions on Evolutionary Computation*, 18(4):519–542.

[Ke et al., 2013] Ke, L., Zhang, Q. e Battiti, R. (2013). MOEA/D-ACO: a multiobjective evolutionary algorithm using decomposition and antcolony. *IEEE transactions on cybernetics*, 43(6):1845–1859.

[Ke et al., 2014] Ke, L., Zhang, Q. e Battiti, R. (2014). Hybridization of decomposition and local search for multiobjective optimization. *IEEE transactions on cybernetics*, 44(10):1808–1820.

[Khan, 2014] Khan, I. H. (2014). A comparative study of EAG and PBIL on large-scale global optimization problems. *Applied Computational Intelligence and Soft Computing*, 2014:1–11.

[Khan, 2003] Khan, N. (2003). Bayesian optimization algorithms for multi-objective and hierarchically difficult problems. Dissertação de Mestrado, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL.

[Knowles e Corne, 2003] Knowles, J. e Corne, D. (2003). Instance generators and test suites for the multiobjective quadratic assignment problem. In *Evolutionary Multi-criterion Optimization*, pages 295–310. Springer.

[Knowles e Corne, 2005] Knowles, J. e Corne, D. (2005). Memetic algorithms for multiobjective optimization: issues, methods and prospects. In *Recent advances in memetic algorithms*, pages 313–352. Springer.

[Knowles et al., 2006] Knowles, J., Thiele, L. e Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report TIK-Report 214, Computer Engineering and Networks Laboratory ETH Zurich Switzerland, Zurich, Switzerland.

[Kochenberger et al., 2014] Kochenberger, G., Hao, J.-K., Glover, F., Lewis, M., Lü, Z., Wang, H. e Wang, Y. (2014). The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28(1):58–81.

[Kong e Tian, 2006] Kong, M. e Tian, P. (2006). Introducing a binary ant colony optimization. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 444–451. Springer.

[Larrañaga et al., 2012] Larrañaga, P., Karshenas, H., Bielza, C. e Santana, R. (2012). A review on probabilistic graphical models in evolutionary computation. *Journal of Heuristics*, 18(5):795–819.

[Larrañaga e Lozano, 2002] Larrañaga, P. e Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer Science & Business Media.

[Laumanns e Ocenasek, 2002] Laumanns, M. e Ocenasek, J. (2002). Bayesian optimization algorithm for multi-objective optimization. In *Proceedings of the 7th Parallel Problem Solving from Nature Conference (PPSN)*, volume 2439 de *Lecture Notes in Computer Science*, pages 298–307. Springer.

[Li et al., 2011] Li, H., Kwong, S. e Hong, Y. (2011). The convergence analysis and specification of the Population-Based Incremental Learning algorithm. *Neurocomputing*, 74(11):1868–1873.

[Li e Zhang, 2009] Li, H. e Zhang, Q. (2009). Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302.

[Li et al., 2015] Li, K., Deb, K., Zhang, Q. e Kwong, S. (2015). An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation*, 19(5):694–716.

[Li e Li, 2015] Li, X. e Li, M. (2015). Multiobjective local search algorithm-based decomposition for multiobjective permutation flow shop scheduling problem. *IEEE Transactions on Engineering Management*, 62(4):544–557.

[Liefooghe et al., 2014] Liefooghe, A., Verel, S. e Hao, J.-K. (2014). A hybrid metaheuristic for multiobjective unconstrained binary quadratic programming. *Applied Soft Computing*, 16:10–19.

[Liefooghe et al., 2015] Liefooghe, A., Vérel, S., Paquete, L. e Hao, J. (2015). Experiments on local search for bi-objective unconstrained binary quadratic programming. In *Evolutionary Multi-Criterion Optimization - 8th International Conference, EMO 2015, Guimarães, Portugal, March 29 -April 1, 2015. Proceedings, Part I*, pages 171–186.

[Liu e Reeves, 2001] Liu, J. e Reeves, C. R. (2001). Constructive and composite heuristic solutions to the Ci scheduling problem. *European Journal of Operational Research*, 132(2):439–452.

[López-Ibánez e Stutzle, 2012] López-Ibánez, M. e Stutzle, T. (2012). The automatic design of multiobjective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 16(6):861–875.

[Lozano, 2000] Lozano, J. A. (2000). Analyzing the population based incremental learning algorithm by means of discrete dynamical systems. *Complex Systems*, 12:465–479.

[Luke, 2009] Luke, S. (2009). *Essentials of metaheuristics*. Lulu Com.

[Ma et al., 2016] Ma, W., Wu, Y. e Yun, J. (2016). Multi-objective Population-Based Incremental Learning for commmunity structure detection. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 537–544. IEEE.

[Mahnig e Mühlenbein, 2001] Mahnig, T. e Mühlenbein, H. (2001). Optimal mutation rate using bayesian priors for estimation of distribution algorithms. In *International Symposium on Stochastic Algorithms*, pages 33–48. Springer.

[Mallows, 1957] Mallows, C. L. (1957). NON-NULL Ranking Models. I. *Biometrika*, 44:114–130.

[Marquet et al., 2014] Marquet, G., Derbel, B., Liefooghe, A. e El-Ghazali, T. (2014). Shake them all! rethinking selection and replacement in MOEA/D. In *XIII International Conference on Parallel Problem Solving from Nature (PPSN)*, volume 8672, pages 641–651. Springer.

[Martins et al., 2011] Martins, J. P., Soares, A. H. M., Vargas, D. V. e Delbem, A. C. B. (2011). Multi-objective phylogenetic algorithm: Solving multi-objective decomposable deceptive problems. In *Evolutionary Multi-Criterion Optimization*, pages 285–297. Springer.

[McCall et al., 2008] McCall, J., Petrovski, A. e Shakya, S. (2008). Evolutionary algorithms for cancer chemotherapy optimization. *Computational Intelligence in Bioinformatics*, pages 265–296.

[Mendiburu et al., 2006] Mendiburu, A., Miguel-Alonso, J., Lozano, J. A., Ostra, M. e Ubide, C. (2006). Parallel EDAs to create multivariate calibration models for quantitative chemical applications. *Journal of Parallel Distributed Computation*, 66(8):1002–1013.

[Mendiburu et al., 2012] Mendiburu, A., Santana, R. e Lozano, J. A. (2012). Fast fitness improvements in estimation of distribution algorithms using belief propagation. In Santana, R. e Shakya, S., editores, *Markov Networks in Evolutionary Computation*, pages 141–155. Springer.

[Miettinen, 2012] Miettinen, K. (2012). *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.

[Minella et al., 2008] Minella, G., Ruiz, R. e Ciavotta, M. (2008). A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *European Journal of Operational Research*, 20(3):451–471.

[Minella et al., 2011] Minella, G., Ruiz, R. e Ciavotta, M. (2011). Restarted Iterated Pareto Greedy algorithm for multi-objective flowshop scheduling problems. *Computers & Operations Research*, 38(11):1521–1533.

[Mohan e Baskaran, 2012] Mohan, B. C. e Baskaran, R. (2012). A survey: Ant colony optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications*, 39(4):4618–4627.

[Mora et al., 2013] Mora, A. M., García-Sánchez, P., Merelo, J. e Castillo, P. A. (2013). Pareto-based multi-colony multi-objective ant colony optimization algorithms: an island model proposal. *Soft Computing*, 17(7):1175–1207.

[Mora et al., 2011] Mora, A. M., Merelo, J., Castillo, P. A., Arenas, M. G., García-Sánchez, P., Laredo, J. L. J. e Romero, G. (2011). A study of parallel approaches in MOACOs for solving the bicriteria TSP. In *International Work-Conference on Artificial Neural Networks*, pages 316–324. Springer.

[Mühlenbein, 1997] Mühlenbein, H. (1997). The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346.

[Mühlenbein et al., 2003] Mühlenbein, H., Mahnig, T. e Ais, F. (2003). Evolutionary algorithms and the boltzmann distribution. In *Foundations of Genetic Algorithms 7*, pages 525–556, San Mateo, CA, USA.

[Mühlenbein et al., 1999] Mühlenbein, H., Mahnig, T. e Ochoa, A. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):213–247.

[Mühlenbein e Paass, 1996] Mühlenbein, H. e Paass, G. (1996). From recombination of genes to the estimation of distributions i. binary parameters. In *International Conference on Parallel Problem Solving from Nature (PPSN)*, pages 178–187. Springer.

[Murata e Ishibuchi, 1994] Murata, T. e Ishibuchi, H. (1994). Performance evaluation of genetic algorithms for flowshop scheduling problems. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 812–817. IEEE.

[Nawaz et al., 1983] Nawaz, M., Enscore, E. E. e Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95.

[Nebro e Durillo, 2010a] Nebro, A. J. e Durillo, J. J. (2010a). A study of the parallelization of the multi-objective metaheuristic MOEA/D. In *International Conference on Learning and Intelligent Optimization*, pages 303–317. Springer.

[Nebro e Durillo, 2010b] Nebro, A. J. e Durillo, J. J. (2010b). A study of the parallelization of the multi-objective metaheuristic MOEA/D. In *Learning and Intelligent Optimization*, pages 303–317. Springer.

[Osman e Laporte, 1996] Osman, I. H. e Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations research*, 63(5):511–623.

[Papadimitriou e Steiglitz, 1982] Papadimitriou, C. H. e Steiglitz, K. (1982). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

[Pareto, 1964] Pareto, V. (1964). *Cours d'économie politique*, volume 1. Librairie Droz.

[Pelikan, 2011] Pelikan, M. (2011). Analysis of epistasis correlation on NK landscapes with nearest neighbor interactions. In *Proceedings of the 13th conference on Genetic and evolutionary computation (GECCO)*, pages 1013–1020. ACM.

[Pelikan et al., 1999] Pelikan, M., Goldberg, D. E. e Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, volume I, pages 525–532, Orlando, FL. Morgan Kaufmann Publishers, San Francisco, CA.

[Pelikan e Mühlenbein, 1999] Pelikan, M. e Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In Roy, R., Furuhashi, T. e Chawdhry, P., editores, *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 521–535, London. Springer.

[Pelikan et al., 2007] Pelikan, M., Sastry, K. e Cantú-Paz, E. (2007). *Scalable optimization via probabilistic modeling: From algorithms to applications*, volume 33. Springer.

[Pelikan et al., 2005] Pelikan, M., Sastry, K. e Goldberg, D. E. (2005). Multiobjective hboa, clustering, and scalability. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 663–670. ACM.

[Pelikan et al., 2006] Pelikan, M., Sastry, K. e Goldberg, D. E. (2006). Multiobjective estimation of distribution algorithms. In *Scalable optimization via probabilistic modeling*, pages 223–248. Springer.

[Pilat e Neruda, 2015] Pilat, M. e Neruda, R. (2015). Incorporating user preferences in MOEA/D through the coevolution of weights. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 727–734. ACM.

[Qi et al., 2014] Qi, Y., Ma, X., Liu, F., Jiao, L., Sun, J. e Wu, J. (2014). MOEA/D with adaptive weight adjustment. *Evolutionary computation*, 22(2):231–264.

[Rastegar, 2011] Rastegar, R. (2011). On the optimal convergence probability of univariate estimation of distribution algorithms. *Evolutionary computation*, 19(2):225–248.

[Rastegar e Hariri, 2006] Rastegar, R. e Hariri, A. (2006). The population-based incremental learning algorithm converges to local optima. *Neurocomputing*, 69(13):1772–1775.

[Rothlauf, 2011] Rothlauf, F. (2011). *Design of modern heuristics: principles and application*. Springer Science & Business Media.

[Salvá et al., 2013] Salvá, T., Emmendorfer, L. R. e Werhli, A. V. (2013). Inference of genetic regulatory networks using an estimation of distribution algorithm. In *Advances in Bioinformatics and Computational Biology*, pages 148–159. Springer.

[Santana et al., 2009] Santana, R., Bielza, C., Lozano, J. A. e Larrañaga, P. (2009). Mining probabilistic models learned by EDAs in the optimization of multi-objective problems. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 445–452. ACM.

[Santana et al., 2008] Santana, R., Larrañaga, P. e Lozano, J. A. (2008). Protein folding in simplified models with estimation of distribution algorithms. *IEEE transactions on Evolutionary Computation*, 12(4):418–438.

[Santana et al., 2012] Santana, R., Mendiburu, A. e Lozano, J. A. (2012). Structural transfer using EDAs: An application to multi-marker tagging SNP selection. In *Proceedings of the 2012 Congress on Evolutionary Computation (CEC)*, pages 3484–3491. IEEE Press.

[Santana et al., 2015] Santana, R., Mendiburu, A. e Lozano, J. A. (2015). Computing factorized approximations of Pareto-fronts using mNM-landscapes and Boltzmann distributions. *CoRR*, abs/1512.03466.

[Santana et al., 2001a] Santana, R., Ochoa, A. e Soto, M. R. (2001a). Factorized Distribution Algorithms for functions with unitation constraints. In *Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems*, pages 158–165.

[Santana et al., 2001b] Santana, R., Ochoa, A. e Soto, M. R. (2001b). The mixture of trees factorized distribution algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 543–550, San Francisco, CA. Morgan Kaufmann Publishers.

[Sastry et al., 2005] Sastry, K., Abbass, H. A., Goldberg, D. E. e Johnson, D. (2005). Substructural niching in estimation of distribution algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 671–678. ACM.

[Sato, 2015] Sato, H. (2015). MOEA/D using constant-distance based neighbors designed for many-objective optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 2867–2874. IEEE.

[Schütze et al., 2012] Schütze, O., Esquivel, X., Lara, A. e Coello, C. A. C. (2012). Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 16(4):504–522.

[Sebag e Ducoulombier, 1998] Sebag, M. e Ducoulombier, A. (1998). Extending population-based incremental learning to continuous search spaces. In *Parallel Problem Solving from Nature - PPSN V*, pages 418–427, Berlin Heidelberg. Springer. Lecture Notes in Computer Science 1498.

[Sheskin, 2003] Sheskin, D. J. (2003). *Handbook of parametric and nonparametric statistical procedures*, volume 3. Chapmanz & Hall CRC.

[Shim et al., 2012] Shim, V. A., Tan, K. C. e Tan, K. K. (2012). A Hybrid Estimation of Distribution Algorithm for Solving the Multi-objective Multiple Traveling Salesman Problem. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

[Smith et al., 2004] Smith, K. I., Everson, R. M. e Fieldsend, J. E. (2004). Dominance measures for multi-objective simulated annealing. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, volume 1, pages 23–30. IEEE.

[Stützle, 1998] Stützle, T. (1998). Local search algorithms for combinatorial problems. *Darmstadt University of Technology PhD Thesis*, 20.

[Stützle e Hoos, 2000] Stützle, T. e Hoos, H. H. (2000). Max–min ant system. *Future generation computer systems*, 16(8):889–914.

[Taillard, 1993] Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285.

[Tan et al., 2013] Tan, Y.-y., Jiao, Y.-c., Li, H. e Wang, X.-k. (2013). MOEA/D+ uniform design: A new version of MOEA/D for optimization problems with many objectives. *Computers & Operations Research*, 40(6):1648–1660.

[Thierens e Bosman, 2001] Thierens, D. e Bosman, P. A. (2001). Multi-objective optimization with iterated density estimation evolutionary algorithms using mixture models. In *Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001), Cuba*, pages 129–136, Havana, Cuba.

[Trivedi et al., 2016] Trivedi, A., Srinivasan, D., Sanyal, K. e Ghosh, A. (2016). A survey of multi-objective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*. Accpeted for publication.

[Uchida et al., 2012] Uchida, A., Ito, Y. e Nakano, K. (2012). An efficient gpu implementation of ant colony optimization for the traveling salesman problem. In *Third International Conference on Networking and Computing*, pages 94–102.

[Valdez et al., 2013] Valdez, S. I., Hernández, A. e Botello, S. (2013). A boltzmann based estimation of distribution algorithm. *Information Sciences*, 236:126–137.

[Ventresca e Tizhoosh, 2008] Ventresca, M. e Tizhoosh, H. R. (2008). A diversity maintaining population-based incremental learning algorithm. *Information Sciences*, 178(21):4038–4056.

[Verel et al., 2013] Verel, S., Liefooghe, A., Jourdan, L. e Dhaenens, C. (2013). On the structure of multiobjective combinatorial search space:MNK-landscapes with correlated objectives. *European Journal of Operational Research*, 227(2):331–342.

[Verel et al., 2011] Verel, S., Ochoa, G. e Tomassini, M. (2011). Local optima networks of NK landscapes with neutrality. *Evolutionary Computation, IEEE Transactions on*, 15(6):783–797.

[Wang et al., 2015] Wang, B., Hua, X. e Yuan, Y. (2015). Scale Adaptive Reproduction Operator for Decomposition based Estimation of Distribution Algorithm. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

[Wang et al., 2014] Wang, Z., Zhang, Q., Gong, M. e Zhou, A. (2014). A replacement strategy for balancing convergence and diversity in MOEA/D. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2132–2139. IEEE.

[Whitley, 2015] Whitley, D. (2015). Mk landscapes, NK landscapes, MAX-kSAT: A proof that the only challenging problems are deceptive. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference (GECCO)*, pages 927–934. ACM.

[Yang e Yao, 2005] Yang, S. e Yao, X. (2005). Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, 9(11):815–834.

[Yenisey e Yagmahan, 2014] Yenisey, M. M. e Yagmahan, B. (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, 45:119–135.

[Yuan e Gallagher, 2003] Yuan, B. e Gallagher, M. (2003). Playing in continuous spaces: Some analysis and extension of population-based incremental learning. In Sarker, R., Reynolds, R., Abbass, H., Tan, K. C., McKay, B., Essam, D. e Gedeon, T., editores, *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 443–450. IEEE Press.

[Zangari e Pozo, 2014a] Zangari, M. e Pozo, A. (2014a). Parallel MOEA/D-ACO on GPU. In *Advances in Artificial Intelligence–IBERAMIA 2014*, pages 405–417. Springer.

[Zangari e Pozo, 2015] Zangari, M. e Pozo, A. (2015). Multiobjective Binary ACO for Unconstrained Binary Quadratic Programming. In *Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS)*, pages 86–91. IEEE.

[Zangari et al., 2016a] Zangari, M., Pozo, A., Santanta, R. e Mendiburu, A. (2016a). A decomposition-based binary ACO algorithm for the multiobjective UBQP. *Neurocomputing*. Accept for publication.

[Zangari e Pozo, 2014b] Zangari, M. e Pozo, A. T. R. (2014b). A GPU Implementation of MOEA/D-ACO for the Multiobjective Traveling Salesman Problem. In *Brazilian Conference on Intelligent Systems (BRACIS)*, pages 324–329. IEEE.

[Zangari et al., 2015a] Zangari, M., Santana, R., Mendiburu, A. e Pozo, A. (2015a). PBIL: un mismo nombre para distintos algoritmos. un caso de estudio sobre un problema de optimización multi-objetivo. In *Proceedings of the Spanish Asociation in Artificial Inteligence (CAEPIA)*, pages 283–295. Accepted for publication.

[Zangari et al., 2016b] Zangari, M., Santana, R., Mendiburu, A. e Pozo, A. (2016b). On the design of hard mUBQP instances. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 421–428, New York, NY, USA. ACM.

[Zangari et al., 2017] Zangari, M., Santana, R., Mendiburu, A. e Pozo, A. (2017). Not all PBILs are the same: Unveiling the different learning mechanisms of PBIL variants. *Applied Soft Computing*, 53:88 – 96.

[Zangari et al., 2015b] Zangari, M., Santana, R. e Pozo, M. A. (2015b). MOEA/D-GM: Using probabilistic graphical models in MOEA/D for solving combinatorial optimization problems. Technical Report arXiv:1511.05625, Department of Computer Science and Artificial Intelligence, University of the Basque Country.

[Zapotecas-Martínez et al., 2015] Zapotecas-Martínez, S., Derbel, B., Liefooghe, A., Brockhoff, D., Aguirre, H. e Tanaka, K. (2015). Injecting CMA-ES into MOEA/D. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 783–790. ACM.

[Zhang e Li, 2007] Zhang, Q. e Li, H. (2007). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transaction Evolutionary Computation*, 11(6):712–731.

[Zhang et al., 2008] Zhang, Q., Zhou, A. e Jin, Y. (2008). RM-MEDA: A regularity model based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 12(1):41–63.

[Zhou et al., 2013a] Zhou, A., Gao, F. e Zhang, G. (2013a). A decomposition based estimation of distribution algorithm for multiobjective traveling salesman problems. *Computers & Mathematics with Applications*, 66(10):1857–1868.

[Zhou et al., 2013b] Zhou, Y., Wang, J. e Yin, J. (2013b). A directional-biased tabu search algorithm for multi-objective unconstrained binary quadratic programming problem. In *Advanced Computational Intelligence (ICACI), 2013 Sixth International Conference on*, pages 281–286. IEEE.

[Zitzler et al., 2008] Zitzler, E., Knowles, J. e Thiele, L. (2008). Quality assessment of pareto set approximations. In *Multiobjective Optimization*, pages 373–404. Springer.

[Zitzler et al., 2001] Zitzler, E., Laumanns, M. e Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Relatório técnico, Technical Report 103, Swiss Federal Institute of Technology.

[Zitzler e Thiele, 1999] Zitzler, E. e Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271.

[Zitzler et al., 2003] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M. e Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE transactions on evolutionary computation*, 7(2):117–132.

Appendix A

MoPFSP: Supplementary results

This appendix presents additional studies regarding the MoPFSP minimizing three combinations of objectives: (i) makespan and total flowtime ($F(\sigma) = \{C_{max}, TFT\}$), (ii) makespan and total tardiness ($F(\sigma) = \{C_{max}, TT\}$), and (iii) makespan, total flowtime and total tardiness ($F(\sigma) = \{C_{max}, TFT, TT\}$).

A.1 Results on $F(\sigma) = \{C_{max}, TFT\}$

Table A.1 (columns 2 and 3) presents the average *HV* results obtained by MOEA/D and MEDA/D-MK using the *Weighted Sum* scalarizing function after $n \times 1000$ generations on the 110 Taillard test instances (Ta_001,...,Ta_110) optimizing the objectives C_{max} and TFT. The best ranked results according to the *Kruskal-Wallis* test (at 5% of significance level) are highlighted in boldface.

The results show that, overall, MEDA/D-MK significantly outperforms MOEA/D in 84 of the 110 test instances. The algorithms does not have a significant difference mainly for the largest test instances (Ta_91,...,Ta_110). These results complement those in Section 7.4.3.

Table A.1: Average HV results for the combination of objectives (i) $F(\sigma) = \{C_{max}, TFT\}$ (columns 2 and 3) and (ii) $F(\sigma) = \{C_{max}, TT\}$ (columns 4 and 5)

Instance	$F(\sigma) = \{C_{max}, TFT\}$		$F(\sigma) = \{C_{max}, TT\}$	
	MOEA/D	MEDA/D-MK	MOEA/D	MEDA/D-MK
Ta_001	0.8722	**0.9276**	**0.7710**	**0.7712**
Ta_002	0.7280	**0.7743**	**0.6841**	**0.7457**
Ta_003	0.7111	**0.8775**	**0.7500**	**0.7794**
Ta_004	**0.7725**	**0.8343**	0.8103	**0.8434**
Ta_005	**0.7440**	**0.7560**	**0.7601**	**0.7744**
Ta_006	0.7344	**0.7440**	**0.5544**	**0.5790**
Ta_007	**0.7312**	**0.7681**	0.7054	**0.7633**
Ta_008	0.8256	**0.9002**	0.9185	**0.9521**
Ta_009	0.6999	**0.7811**	0.7990	**0.8339**
Ta_010	0.7637	**0.8116**	0.7805	**0.8241**
Ta_011	0.6409	**0.7531**	0.6541	**0.7140**
Ta_012	0.6316	**0.7510**	0.7532	**0.7806**
Ta_013	0.6858	**0.8189**	0.7859	**0.8212**
Ta_014	0.7121	**0.8012**	0.7538	**0.7947**
Ta_015	**0.6966**	**0.8572**	0.7503	**0.8235**
Ta_016	0.7929	**0.8553**	0.6653	**0.6959**

Continued on next page

Table A.1 – continued from previous page

Instance	$F(\sigma) = \{C_{max}, TFT\}$		$F(\sigma) = \{C_{max}, TT\}$	
	MOEA/D	MEDA/D-MK	MOEA/D	MEDA/D-MK
Ta_017	0.7704	**0.7828**	0.5157	**0.6287**
Ta_018	0.8037	**0.9035**	0.6462	**0.8618**
Ta_019	0.7302	**0.8448**	0.7773	**0.8156**
Ta_020	0.7432	**0.8248**	0.7602	**0.8032**
Ta_021	0.5342	**0.6794**	**0.5897**	**0.6227**
Ta_022	0.6870	**0.8121**	0.7724	**0.7981**
Ta_023	0.6576	**0.7370**	0.6500	**0.7731**
Ta_024	0.8710	**0.9234**	0.5068	**0.7317**
Ta_025	**0.6588**	**0.6884**	0.7434	**0.7923**
Ta_026	0.7108	**0.7780**	**0.7878**	**0.8097**
Ta_027	0.7194	**0.8262**	0.6791	**0.7716**
Ta_028	0.6665	**0.7503**	0.7007	**0.7386**
Ta_029	0.7037	**0.8000**	0.6199	**0.6732**
Ta_030	0.6306	**0.7723**	0.5472	**0.5910**
Ta_031	**0.8858**	**0.9184**	0.6885	**0.8953**
Ta_032	0.6935	**0.7586**	0.6301	**0.8055**
Ta_033	0.7723	**0.8501**	0.7868	**0.9267**
Ta_034	0.6467	**0.7921**	0.7117	**0.9073**
Ta_035	0.7974	**0.9369**	0.7775	**0.9700**
Ta_036	0.6348	**0.7352**	0.7591	**0.9082**
Ta_037	0.7619	**0.8460**	0.7663	**0.9200**
Ta_038	0.8657	**0.9227**	0.8135	**0.9326**
Ta_039	0.5733	**0.6997**	0.6947	**0.8708**
Ta_040	0.7914	**0.9082**	0.7672	**0.9360**
Ta_041	0.6197	**0.6974**	0.6700	**0.8038**
Ta_042	0.5198	**0.6344**	0.7388	**0.8651**
Ta_043	0.5331	**0.6028**	0.6515	**0.7902**
Ta_044	0.6896	**0.7347**	0.6720	**0.8599**
Ta_045	**0.6030**	**0.6706**	0.6521	**0.7890**
Ta_046	0.5808	**0.6557**	0.7041	**0.8269**
Ta_047	0.6336	**0.6745**	0.7617	**0.8838**
Ta_048	0.5948	**0.6572**	0.7123	**0.8676**
Ta_049	0.6390	**0.6996**	0.7368	**0.8248**
Ta_050	0.5243	**0.6008**	0.6575	**0.7750**
Ta_051	0.5699	**0.6622**	0.7242	**0.8128**
Ta_052	**0.5726**	**0.5603**	0.6625	**0.7539**
Ta_053	0.5645	**0.6671**	0.7491	**0.8286**
Ta_054	0.5419	**0.6592**	0.7501	**0.8488**
Ta_055	0.4919	**0.6031**	0.6424	**0.7370**
Ta_056	0.5583	**0.6611**	0.7886	**0.8438**
Ta_057	0.4625	**0.6252**	0.7588	**0.8279**
Ta_058	0.6331	**0.7067**	0.6989	**0.8062**
Ta_059	0.6033	**0.6560**	0.7400	**0.7902**
Ta_060	0.7125	**0.7789**	0.8117	**0.8630**
Ta_061	0.7784	**0.8738**	0.7713	**0.9056**
Ta_062	**0.8442**	**0.8611**	0.4999	**0.7537**
Ta_063	0.6537	0.7101	0.7235	**0.8420**
Ta_064	0.7838	0.8806	0.7210	**0.8814**
Ta_065	0.8312	0.9007	0.8692	**0.9552**
Ta_066	0.7369	0.7679	0.6499	**0.8267**
Ta_067	0.6861	**0.7993**	0.6383	**0.8148**
Ta_068	0.5837	**0.7111**	0.5558	**0.7504**
Ta_069	0.6609	**0.7227**	0.3585	**0.5974**

Continued on next page

Table A.1 – continued from previous page

Instance	$F(\sigma) = \{C_{max}, TFT\}$		$F(\sigma) = \{C_{max}, TT\}$	
	MOEA/D	MEDA/D-MK	MOEA/D	MEDA/D-MK
Ta_070	0.7112	**0.8134**	0.7452	**0.8835**
Ta_071	0.5507	**0.6120**	0.7036	**0.8297**
Ta_072	**0.6600**	**0.6898**	0.7179	**0.8391**
Ta_073	0.5284	**0.6124**	0.5891	**0.7553**
Ta_074	0.5164	**0.5960**	0.5913	**0.7979**
Ta_075	0.5193	**0.6018**	0.8108	**0.8997**
Ta_076	0.4833	**0.6150**	0.6005	**0.7764**
Ta_077	0.4942	**0.5613**	0.6583	**0.7497**
Ta_078	0.4561	**0.5719**	0.5946	**0.7374**
Ta_079	0.4555	**0.5279**	0.6623	**0.7882**
Ta_080	0.6270	**0.7019**	0.6715	**0.7915**
Ta_081	0.4645	**0.5407**	0.7388	**0.8388**
Ta_082	0.3876	**0.4958**	0.6542	**0.7837**
Ta_083	0.4457	**0.5372**	0.6118	**0.7923**
Ta_084	0.4276	**0.4915**	0.6582	**0.7567**
Ta_085	0.4838	**0.5634**	0.5703	**0.7780**
Ta_086	**0.4672**	**0.5101**	0.7132	**0.8303**
Ta_087	**0.4670**	**0.4891**	0.5440	**0.7075**
Ta_088	0.4555	0.5317	0.5516	**0.7598**
Ta_089	0.5800	0.5665	0.6221	**0.8185**
Ta_090	0.6115	**0.6435**	0.6975	**0.7772**
Ta_091	0.5965	**0.6934**	**0.7151**	**0.7666**
Ta_092	**0.5612**	**0.5728**	0.7555	**0.8267**
Ta_093	**0.5798**	**0.5997**	0.7060	**0.7866**
Ta_094	**0.6793**	**0.6512**	0.6906	**0.7868**
Ta_095	**0.4965**	**0.5393**	0.7411	**0.8022**
Ta_096	**0.5846**	**0.6101**	**0.7481**	**0.8382**
Ta_097	**0.5687**	**0.6090**	0.7962	**0.8534**
Ta_098	0.4535	**0.5066**	0.7382	**0.8272**
Ta_099	0.5325	**0.6161**	0.6868	**0.8280**
Ta_100	**0.4528**	**0.4627**	**0.5209**	**0.5813**
Ta_101	**0.4480**	**0.4422**	0.6384	**0.7212**
Ta_102	0.4035	**0.4713**	0.5535	**0.7060**
Ta_103	**0.4708**	**0.5062**	0.6180	**0.7808**
Ta_104	0.4473	**0.5238**	0.5744	**0.6852**
Ta_105	0.3811	**0.4438**	0.5083	**0.6337**
Ta_106	**0.4561**	**0.4811**	**0.6375**	**0.6969**
Ta_107	**0.3703**	**0.4182**	0.6328	**0.7193**
Ta_108	**0.4738**	**0.4960**	0.5397	**0.6843**
Ta_109	**0.3492**	**0.4051**	0.6439	**0.7299**
Ta_110	**0.4468**	**0.4520**	0.5100	**0.6939**

The execution time analysis was performed on a PC with Intel Xeon E5-620 2.4 GHz processor and 12 GB memory. Table A.2 indicates that MEDA/D-MK consumes more CPU time than MOEA/D. This is because the Mallows Model EDA components, learning and sampling steps, involve more computational overhead than crossover and mutation operators. Regarding that, the computational cost of the sampling in the MEDA/D-MK is $O(n^2)$. This should explain why the difference between both algorithms increases as the problem size increases.

Table A.2: Average CPU time (in seconds) used by MOEA/D^T and MEDA/D^T

Taillard Instance	MOEA/D		MEDA/D-MK	
	mean	std ved	mean	std dev
Ta_001	73.05	8.83	93.15	2.84
Ta_002	72.19	6.59	92.33	1.46
Ta_003	79.95	4.34	91.09	4.05
Ta_004	73.04	5.00	92.89	3.45
Ta_005	74.96	3.14	94.97	4.18
Ta_006	75.73	7.71	91.86	2.42
Ta_007	79.01	3.29	93.39	2.14
Ta_008	72.07	5.40	91.31	2.20
Ta_009	71.93	5.60	93.37	4.30
Ta_010	72.37	4.42	92.02	3.78
Ta_011	89.59	6.89	101.42	1.88
Ta_012	83.29	5.76	102.44	3.13
Ta_013	81.65	3.54	96.30	5.18
Ta_014	84.19	1.60	103.58	2.19
Ta_015	85.60	2.93	103.31	1.50
Ta_016	87.04	3.31	102.10	2.60
Ta_017	87.01	4.94	97.61	3.42
Ta_018	83.59	1.42	98.93	1.70
Ta_019	80.57	3.50	103.25	2.09
Ta_020	88.45	2.90	101.50	2.23
Ta_021	100.51	3.47	117.04	3.62
Ta_022	99.47	2.23	118.18	1.90
Ta_023	100.36	3.12	121.18	2.53
Ta_024	101.67	2.25	117.09	2.45
Ta_025	105.48	3.75	117.78	2.34
Ta_026	103.58	2.74	115.64	2.25
Ta_027	101.43	2.56	119.57	3.51
Ta_028	99.38	4.37	118.92	3.08
Ta_029	100.44	4.75	118.43	2.62
Ta_030	105.53	3.81	116.99	2.37
Ta_031	217.35	6.43	303.42	5.98
Ta_032	228.03	5.97	312.20	2.89
Ta_033	224.24	6.10	307.18	7.81
Ta_034	221.43	5.89	305.71	4.49
Ta_035	223.08	3.24	309.09	4.70
Ta_036	225.31	6.49	310.94	3.52
Ta_037	222.35	3.13	306.17	4.75
Ta_038	221.72	7.12	309.08	4.10
Ta_039	228.40	6.00	305.58	8.07
Ta_040	223.07	2.63	305.41	4.81
Ta_041	271.91	3.51	368.68	4.41
Ta_042	279.66	6.22	366.26	3.21
Ta_043	277.87	2.01	366.54	2.19
Ta_044	278.06	5.95	366.01	6.25
Ta_045	278.01	4.37	366.33	4.98
Ta_046	282.20	9.51	369.12	5.05
Ta_047	275.67	4.01	362.17	4.25
Ta_048	279.91	3.59	368.40	2.97
Ta_049	279.47	4.64	367.16	4.00
Ta_050	276.11	9.23	367.18	8.11
Ta_051	389.90	3.72	480.24	2.74
Ta_052	393.41	3.12	482.20	2.18

Continued on next page

Table A.2 – continued from previous page

Taillard Instance	MOEA/D		MEDA/D-MK	
	mean	std ved	mean	std dev
Ta_053	389.67	4.35	478.65	2.48
Ta_054	389.59	4.45	476.73	3.43
Ta_055	399.29	5.85	480.22	4.41
Ta_056	394.70	5.63	481.69	5.48
Ta_057	394.38	5.64	477.36	6.18
Ta_058	392.12	5.31	477.43	3.46
Ta_059	396.09	7.70	478.83	4.29
Ta_060	391.73	3.22	477.30	8.15
Ta_061	548.77	6.86	892.75	9.54
Ta_062	564.76	14.99	880.62	9.46
Ta_063	547.39	6.97	883.68	8.30
Ta_064	549.89	8.52	892.55	13.22
Ta_065	561.64	12.25	867.95	10.37
Ta_066	552.11	5.90	869.71	12.62
Ta_067	549.52	5.24	889.50	5.13
Ta_068	557.51	9.90	874.56	15.09
Ta_069	547.54	6.34	884.94	5.10
Ta_070	550.07	9.65	880.50	3.91
Ta_071	775.07	6.93	1118.42	8.04
Ta_072	764.39	7.49	1106.47	6.80
Ta_073	782.00	7.33	1114.78	14.42
Ta_074	762.59	5.81	1114.07	8.07
Ta_075	787.13	9.43	1127.19	5.05
Ta_076	774.33	6.09	1110.00	6.41
Ta_077	772.64	6.65	1125.14	8.99
Ta_078	778.16	4.80	1119.39	10.38
Ta_079	780.24	5.43	1124.94	10.05
Ta_080	765.56	3.09	1123.50	5.14
Ta_081	1241.83	8.96	1586.35	9.08
Ta_082	1242.28	4.55	1586.54	9.09
Ta_083	1243.11	5.30	1598.08	9.31
Ta_084	1232.42	7.14	1573.39	7.75
Ta_085	1251.57	5.90	1589.51	10.26
Ta_086	1246.92	9.88	1585.96	6.85
Ta_087	1232.00	6.84	1575.23	10.91
Ta_088	1253.64	6.89	1593.02	9.83
Ta_089	1245.93	6.68	1586.87	11.24
Ta_090	1241.25	8.28	1587.45	8.55
Ta_091	2491.88	27.21	3789.63	25.26
Ta_092	2477.69	15.23	3760.98	22.98
Ta_093	2492.70	8.00	3786.04	14.15
Ta_094	2457.41	10.34	3787.36	41.81
Ta_095	2482.50	17.15	3749.81	22.27
Ta_096	2480.92	6.24	3739.61	13.52
Ta_097	2461.74	16.93	3792.48	24.49
Ta_098	2471.82	8.56	3731.21	19.35
Ta_099	2450.06	11.49	3724.39	11.58
Ta_100	2469.99	15.26	3754.15	19.43
Ta_101	4293.12	15.47	5639.96	24.15
Ta_102	4311.73	34.07	5584.47	20.77
Ta_103	4286.46	32.44	5571.16	20.14
Ta_104	4244.79	20.87	5544.95	32.86
Ta_105	4295.91	15.80	5600.41	25.71

Continued on next page

Table A.2 – continued from previous page

Taillard Instance	MOEA/D		MEDA/D-MK	
	mean	std ved	mean	std dev
Ta_106	4289.32	12.25	5578.47	23.18
Ta_107	4260.43	11.62	5582.55	15.94
Ta_108	4313.17	33.16	5597.05	28.62
Ta_109	4294.80	13.40	5614.86	29.65
Ta_110	3501.59	607.55	5562.85	23.67

We have produced our reference sets (best-known) for each test instance. Table A.3 presents the *HV* results obtained by MEDA/D-MK compared to the reference best-known sets from Dubois-Lacoste et al. (TP+PLS) [Dubois-Lacoste et al., 2011] and Minella et. al [Minella et al., 2011] for the 110 Taillard test instances optimizing the objectives Total Flowtime (TFT), and the makespan (C_{max}).

Table A.3: Hypervolume obtained from the reference sets

Taillard Instance	TP+PLS	Minella et al.	MEDA/D-MK
20×5			
Ta_001	**0.436**	**0.436**	**0.436**
Ta_002	**0.492**	0.464	0.461
Ta_003	**0.899**	0.894	0.895
Ta_004	**0.861**	0.859	0.849
Ta_005	0.629	**0.644**	**0.644**
Ta_006	**0.589**	**0.589**	0.576
Ta_007	**0.816**	0.792	0.792
Ta_008	0.884	**0.882**	**0.882**
Ta_009	0.785	0.782	**0.785**
Ta_010	0.658	**0.659**	**0.659**
20×10			
Ta_011	0.726	**0.739**	**0.739**
Ta_012	0.766	0.772	**0.774**
Ta_013	0.845	0.846	**0.847**
Ta_014	0.787	0.775	**0.789**
Ta_015	0.612	0.613	**0.614**
Ta_016	**0.850**	0.845	0.844
Ta_017	**0.713**	**0.713**	**0.713**
Ta_018	**0.840**	0.828	**0.840**
Ta_019	**0.943**	**0.943**	**0.943**
Ta_020	0.853	0.849	**0.854**
20×20			
Ta_021	**0.643**	0.628	0.619
Ta_022	0.773	**0.779**	0.778
Ta_023	0.808	0.807	**0.809**
Ta_024	**0.874**	0.864	**0.874**
Ta_025	**0.821**	0.818	**0.821**
Ta_026	0.867	**0.878**	0.876
Ta_027	**0.827**	0.804	0.822
Ta_028	0.824	0.823	**0.826**
Ta_029	**0.767**	**0.767**	**0.767**
Ta_030	0.764	0.777	**0.781**
50×5			
Ta_031	**0.954**	0.771	0.864
Ta_032	**0.953**	0.865	0.911
Ta_033	**0.922**	0.806	0.876

Continued on next page

Table A.3 – continued from previous page

Taillard Instance	TP+PLS	Minella et al.	MEDA/D-MK
Ta_034	**0.976**	0.893	0.942
Ta_035	**0.985**	0.910	0.970
Ta_036	**0.953**	0.909	0.936
Ta_037	**0.832**	0.760	0.752
Ta_038	**0.904**	0.782	0.851
Ta_039	**0.927**	0.880	0.894
Ta_040	**0.964**	0.788	0.912
50×10			
Ta_041	**0.919**	0.702	0.827
Ta_042	**0.802**	0.674	0.764
Ta_043	**0.866**	0.696	0.832
Ta_044	**0.960**	0.788	0.923
Ta_045	**0.890**	0.649	0.855
Ta_046	**0.874**	0.686	0.866
Ta_047	**0.906**	0.731	0.864
Ta_048	**0.879**	0.714	0.770
Ta_049	**0.896**	0.707	0.794
Ta_050	0.912	0.737	0.871
50×20			
Ta_051	**0.906**	0.615	0.832
Ta_052	**0.852**	0.594	0.816
Ta_053	**0.890**	0.612	0.825
Ta_054	**0.863**	0.656	0.810
Ta_055	**0.778**	0.516	0.688
Ta_056	**0.902**	0.708	0.817
Ta_057	**0.888**	0.616	0.801
Ta_058	**0.918**	0.726	0.857
Ta_059	**0.813**	0.585	0.705
Ta_060	**0.913**	0.695	0.860
100×5			
Ta_061	**0.978**	0.749	0.925
Ta_062	**0.985**	0.880	0.843
Ta_063	**0.970**	0.835	0.884
Ta_064	0.964	0.877	**0.966**
Ta_065	**0.992**	0.888	**0.992**
Ta_066	**0.988**	0.896	0.932
Ta_067	0.878	0.654	**0.937**
Ta_068	0.935	0.842	**0.946**
Ta_069	**0.911**	0.664	0.753
Ta_070	**0.962**	0.856	0.943
100×10			
Ta_071	**0.909**	0.762	0.844
Ta_072	**0.933**	0.779	0.856
Ta_073	**0.965**	0.766	0.851
Ta_074	**0.819**	0.564	0.804
Ta_075	**0.942**	0.764	0.793
Ta_076	**0.976**	0.755	0.905
Ta_077	**0.898**	0.745	0.770
Ta_078	0.840	0.679	**0.844**
Ta_079	**0.892**	0.708	0.857
Ta_080	**0.857**	0.610	0.620
100×20			
Ta_081	0.904	0.692	**0.933**
Ta_082	**0.859**	0.534	0.784

Continued on next page

Table A.3 – continued from previous page

Taillard Instance	TP+PLS	Minella et al.	MEDA/D-MK
Ta_083	**0.893**	0.586	0.840
Ta_084	**0.908**	0.652	0.827
Ta_085	**0.879**	0.540	0.847
Ta_086	**0.862**	0.476	0.826
Ta_087	**0.811**	0.484	0.802
Ta_088	0.863	0.556	**0.892**
Ta_089	**0.906**	0.598	0.853
Ta_090	**0.867**	0.547	0.843
200×10			
Ta_091	0.854	0.622	**0.947**
Ta_092	0.823	0.715	**0.903**
Ta_093	0.713	0.637	**0.793**
Ta_094	0.767	0.734	**0.875**
Ta_095	0.901	0.750	**0.967**
Ta_096	0.896	0.768	**0.899**
Ta_097	**0.855**	0.718	0.827
Ta_098	0.836	0.767	**0.882**
Ta_099	0.911	0.818	**0.962**
Ta_100	0.714	0.679	**0.758**
200×20			
Ta_101	0.820	0.611	**0.918**
Ta_102	0.792	0.527	**0.882**
Ta_103	0.845	0.576	**0.901**
Ta_104	0.805	0.580	**0.850**
Ta_105	0.775	0.562	**0.782**
Ta_106	0.802	0.560	**0.914**
Ta_107	0.740	0.450	**0.859**
Ta_108	**0.873**	0.564	0.769
Ta_109	0.862	0.632	**0.918**
Ta_110	**0.858**	0.525	0.845

The results show that MEDA/D-MK achieves competitive results. Our approach achieves the best results, in most of the cases, for the groups $20 \times 10, 200 \times 10$, and 200×20. The best-known sets from [Dubois-Lacoste et al., 2011] achieves the best *HV* values for 50 and 100 jobs. Our best-known approximated Pareto fronts are available on-line[1] for future comparison to other approaches.

A.2 Results on $F(\sigma) = \{C_{max}, TFT, TT\}$

In this section, we present the experimental study minimizing three PFSP objectives simultaneously. Unfortunately, for this study, a set of *best-known* approximated *PFs* for comparison is not available. Therefore, we have compared only MOEA/D and MEDA/D-MK using three different scalarizing function approaches (the *Weighted Sum*, the *Tchebycheff*, and the *PBI*). We set the penalty parameter from *PBI* to $\theta_{PBI} = 5.0$. It means that the *PBI* scalarizing function has a sharp edge, which leads to slower convergence towards the reference point. The ordering of the scalarizing functions in favor of diversity in the *objective space* is: *Weighted Sum* < *Tchebycheff* < *PBI* 5.0.

Moreover, we set the number of subproblems to $N = 231$. As the stopping condition, the algorithms stop after $MaxGen = n \times 1000$ generations. The remaining parameters are set

[1] Available at at https://github.com/MuriloZangari/supplementary_results_mopfsp

as in Chapter 7, i.e., neighborhood size $T = 10$, probability to sample the central permutation $P(\sigma_0) = 0.8$, and maximum number of updates by a new solution $n_r = 2$.

Table A.4: Average normalized HV results for $F(\sigma) = \{C_{max}, TFT, TT\}$

Instance	MOEA/D			MEDA/D-MK		
	WS	*TC*	*PBI*	*WS*	*TC*	*PBI*
Ta_001	0.8958	0.8704	0.7561	**0.8959**	0.886	0.8164
Ta_002	0.8583	0.9011	0.7853	**0.8945**	**0.9120**	0.7964
Ta_003	0.8427	0.8589	0.5494	**0.8733**	0.8655	0.6341
Ta_004	0.8008	0.8201	0.6316	**0.8307**	0.8257	0.6942
Ta_005	0.7953	0.8178	0.6369	**0.8217**	0.8171	0.6638
Ta_006	0.7212	0.7512	0.6167	0.7395	**0.7613**	0.6507
Ta_007	0.8967	0.9142	0.7714	**0.9536**	0.9493	0.7981
Ta_008	0.9274	**0.9482**	0.8108	**0.9584**	**0.9515**	0.9118
Ta_009	0.858	0.8835	0.6884	**0.8976**	0.8888	0.7479
Ta_010	0.7905	0.8096	0.6622	**0.8244**	0.8136	0.6778
Ta_011	0.9109	0.9268	0.6591	**0.9305**	0.9271	0.7458
Ta_012	0.7667	0.7899	0.5456	**0.7949**	0.7814	0.5977
Ta_013	0.8292	0.8501	0.6782	**0.8597**	0.8527	0.7348
Ta_014	0.7898	0.8279	0.6184	**0.8328**	0.8229	0.6835
Ta_015	0.8909	0.9079	0.7286	**0.9282**	**0.9219**	0.7763
Ta_016	0.7755	**0.8021**	0.5598	**0.7955**	**0.8004**	0.5373
Ta_017	0.7449	**0.8179**	0.6298	**0.8145**	**0.8171**	0.7027
Ta_018	0.859	0.9098	0.7561	**0.9198**	**0.9185**	0.8698
Ta_019	0.8298	**0.8619**	0.6174	**0.8569**	**0.8651**	0.6609
Ta_020	0.7998	**0.8379**	0.6284	**0.8551**	**0.8440**	0.7115
Ta_021	0.739	0.788	0.7585	0.7835	**0.8490**	0.7975
Ta_022	0.9094	**0.9176**	0.7326	**0.9184**	**0.9174**	0.7427
Ta_023	0.9646	0.9731	0.9339	**0.9880**	**0.9871**	0.9464
Ta_024	0.7199	0.772	0.7614	**0.8526**	0.8173	0.8564
Ta_025	0.7548	**0.7810**	0.555	**0.7930**	**0.7846**	0.6332
Ta_026	0.799	**0.8121**	0.6053	**0.8162**	**0.8113**	0.5969
Ta_027	0.8764	0.8956	0.6727	**0.9121**	0.8968	0.7688
Ta_028	0.8188	0.8386	0.7204	**0.8494**	0.8456	0.7592
Ta_029	0.8151	**0.8422**	0.7443	**0.8400**	**0.8451**	0.8012
Ta_030	0.8621	0.9183	0.862	0.8901	**0.9404**	0.9225
Ta_031	0.8791	0.8622	0.8344	**0.9379**	0.9248	0.9111
Ta_032	0.7892	0.7996	0.719	**0.8976**	**0.8742**	0.7941
Ta_033	0.8611	0.8172	0.7712	**0.9498**	0.8764	0.8399
Ta_034	0.8424	0.8588	0.7655	**0.9474**	0.9151	0.8852
Ta_035	0.8724	0.8197	0.8126	**0.9773**	0.9338	0.9113
Ta_036	0.8438	0.7778	0.7417	**0.9481**	0.8722	0.8445
Ta_037	0.8126	0.7772	0.7413	**0.9255**	0.8761	0.8587
Ta_038	0.8279	0.8222	0.8077	**0.8732**	**0.8848**	**0.8634**
Ta_039	0.8062	0.7982	0.7147	**0.9126**	**0.8657**	0.8223
Ta_040	0.891	0.8923	0.853	**0.9699**	0.9458	0.9287
Ta_041	0.7262	0.7391	0.5914	**0.8397**	**0.7969**	0.6701
Ta_042	0.7342	0.7237	0.5858	**0.8166**	0.7489	0.6719
Ta_043	0.6842	0.6927	0.524	**0.8035**	0.7172	0.6193
Ta_044	0.7524	0.764	0.5958	**0.8674**	0.8114	0.6958
Ta_045	0.7108	0.7169	0.56	**0.8194**	0.7324	0.6678
Ta_046	0.7697	0.7726	0.658	**0.8528**	0.8061	0.7461
Ta_047	0.7906	0.7812	0.6742	**0.8779**	0.793	0.7565
Ta_048	0.7912	0.7939	0.6488	**0.8731**	0.8087	0.7121
Ta_049	0.7356	0.7761	0.6202	**0.8324**	0.7854	0.7216

Continued on next page

Table A.4 – continued from previous page

Instance	MOEA/D			MEDA/D-MK		
	WS	*TC*	*PBI*	*WS*	*TC*	*PBI*
Ta_050	0.7103	0.7125	0.5945	**0.8165**	0.7301	0.6702
Ta_051	0.782	0.7639	0.5679	**0.8475**	0.7632	0.6532
Ta_052	0.7232	0.7108	0.4783	**0.7952**	0.7072	0.5392
Ta_053	0.8005	0.7815	0.5335	**0.8528**	0.7751	0.6247
Ta_054	0.7761	0.7716	0.5538	**0.8422**	0.7633	0.6235
Ta_055	0.6756	0.6955	0.4659	**0.7511**	0.6868	0.5378
Ta_056	0.8052	0.7945	0.5792	**0.8597**	0.7889	0.6424
Ta_057	0.779	0.7717	0.584	**0.8340**	0.775	0.6509
Ta_058	0.7676	0.7786	0.5573	**0.8402**	0.7784	0.6339
Ta_059	0.7366	0.7352	0.556	**0.8293**	0.7458	0.6624
Ta_060	0.8145	0.8102	0.6611	**0.8637**	0.8003	0.7052
Ta_061	0.9532	0.9044	0.8878	**0.9859**	0.9515	0.9552
Ta_062	0.888	0.8804	0.8332	**0.9152**	0.9074	0.8932
Ta_063	0.8269	0.8475	0.7576	0.873	**0.8998**	0.8069
Ta_064	0.9048	0.876	0.8372	**0.9502**	0.9205	0.8918
Ta_065	0.93	0.9124	0.8547	**0.9739**	0.95	0.9092
Ta_066	0.8339	0.8787	0.774	0.9111	**0.9155**	0.8333
Ta_067	0.8619	0.8537	0.8098	**0.9234**	**0.9065**	0.8599
Ta_068	0.8221	0.7995	0.7509	**0.8852**	**0.8856**	0.8147
Ta_069	0.8186	0.8613	0.779	0.8678	**0.9192**	0.8213
Ta_070	0.8851	0.8554	0.8308	**0.9335**	**0.9124**	0.8822
Ta_071	0.7547	0.7192	0.6172	**0.8092**	0.7656	0.7046
Ta_072	0.8321	0.7799	0.73	**0.8818**	0.8313	0.7948
Ta_073	0.7848	0.7678	0.6913	**0.8293**	0.8061	0.761
Ta_074	0.7143	0.6847	0.6227	**0.7909**	0.7289	0.7066
Ta_075	0.8249	0.7965	0.7099	**0.8629**	0.8341	0.7535
Ta_076	0.7731	0.7037	0.6857	**0.8823**	0.7785	0.7712
Ta_077	0.7524	0.7202	0.6817	**0.8113**	0.786	0.741
Ta_078	0.7322	0.6973	0.6598	**0.8135**	0.7295	0.731
Ta_079	0.6755	0.6531	0.5921	**0.7341**	0.7047	0.6697
Ta_080	0.7176	0.7627	0.6828	0.7396	**0.8166**	0.7144
Ta_081	0.718	0.6596	0.6192	**0.8181**	0.6901	0.7044
Ta_082	0.7573	0.7004	0.6298	**0.8434**	0.7318	0.7155
Ta_083	0.6964	0.6292	0.5835	**0.7801**	0.6559	0.6466
Ta_084	0.7229	0.6448	0.5986	**0.7983**	0.6746	0.6953
Ta_085	0.7233	0.5937	0.6325	**0.8350**	0.6388	0.7447
Ta_086	0.7056	0.6395	0.6052	**0.8076**	0.6586	0.6798
Ta_087	0.6562	0.6264	0.559	**0.7912**	0.6713	0.6725
Ta_088	0.7079	0.6243	0.6041	**0.8260**	0.6562	0.7171
Ta_089	0.7631	0.6699	0.6588	**0.8670**	0.7084	0.7591
Ta_090	0.7698	0.6363	0.6857	**0.8457**	0.6818	0.7695
Ta_091	**0.9723**	0.8847	0.8583	**0.9844**	0.9125	0.8842
Ta_092	0.8669	0.8174	0.783	**0.8927**	0.8395	0.8135
Ta_093	0.7813	0.7096	0.7037	**0.8328**	0.7385	0.743
Ta_094	0.7874	0.7674	0.6797	**0.8178**	0.7742	0.7327
Ta_095	0.8904	0.7473	0.7883	**0.9243**	0.7859	0.8369
Ta_096	0.9092	0.8364	0.8131	**0.9336**	0.8711	0.8446
Ta_097	0.8016	0.7903	0.7436	0.8318	**0.8471**	0.7638
Ta_098	0.8214	0.7509	0.7775	**0.8578**	0.7763	0.8131
Ta_099	0.864	0.7307	0.7832	**0.8894**	0.7831	0.8281
Ta_100	0.6234	0.6642	0.5771	0.6666	**0.7039**	0.6036
Ta_101	0.703	0.586	0.6033	**0.7832**	0.6049	0.6718
Ta_102	0.7093	0.5391	0.5756	**0.7917**	0.5814	0.7111

Continued on next page

Table A.4 – continued from previous page

Instance	MOEA/D			MEDA/D-MK		
	WS	*TC*	*PBI*	*WS*	*TC*	*PBI*
Ta_103	0.7223	0.5563	0.6101	**0.8216**	0.6145	0.7355
Ta_104	0.7434	0.6494	0.6574	**0.8390**	0.6946	0.738
Ta_105	0.7353	0.6431	0.6234	**0.8164**	0.6786	0.6713
Ta_106	0.789	0.5981	0.6796	**0.8659**	0.6293	0.7474
Ta_107	0.776	0.6081	0.6569	**0.8477**	0.6277	0.7465
Ta_108	0.6859	0.5831	0.6283	**0.8095**	0.622	0.703
Ta_109	0.7934	0.6391	0.6652	**0.8638**	0.6759	0.7874
Ta_110	0.7045	0.5889	0.6065	**0.8077**	0.629	0.7265

The results from Table A.4 show that MEDA/D-MK outperforms MOEA/D in most of the cases for minimizing three objectives. Moreover, the *Weighted Sum* keeps being the best scalarizing function in the context of the decomposition approach for solve the MoPFSP. These results indicate that the choice of the scalarizing function has a fundamental impact on the results, which can lead to more convergence or diversity during the search. Moreover, we make our *best-known* results available[2] for further studies. Moreover, the concept of using multiples scalarizing functions simultaneously can be investigated in the future.

[2]https://github.com/MuriloZangari/supplementary_results_mopfsp

Appendix B

Parallel MOEA/D-ACO on GPU

B.1 Introduction

ACO algorithms are, usually, more expansive than EAs. Due to this parallel ACO have been investigated over time. Recently, ACO on GPU (Graphic Processing Unit) have been proposed to solve single-objective problems [Dawson e Stewart, 2013, Delevacq et al., 2013, Cecilia et al., 2013, Uchida et al., 2012].

In the multi-objective case, some studies have proposed parallel CPU-based MOACOs [Mora et al., 2013, Mora et al., 2011]. Also, a parallel CPU-based MOEA/D [Nebro e Durillo, 2010a] was proposed for continuous MOPs. Those studies have not achieved high speedups due to target hardware. Parallel MOACO algorithms on GPU is a recent and open research field.

NVIDIA introduced CUDA (Compute Unified Device Architecture), a general purpose parallel computing platform and programming model for direct execution on GPUs to solve many complex computational problems in a more efficient way than on a CPU. CUDA[1] exposes the GPU's massively parallel architecture so parallel code can be written to execute faster than its optimized sequential counterpart. The success of a GPU approach depends on the nature of the particular problem and the underlying hardware available.

In this appendix, we describe a parallel implementation of MOEA/D-ACO [Ke et al., 2013] on GPU with CUDA, where both stages *solution construction* and the *pheromone update* are parallelized for solving several instances from MOKP and MOTS. The solution quality and the speedups are compared to the sequential counterpart with different number of objectives and subproblems.

B.2 MOEA/D-ACO

MOEA/D-ACO algorithm decomposes a MOP into N single-objective subproblems by choosing N weight vectors $\lambda^1, ..., \lambda^N$. Subproblem i is associated with weight vector λ^i and its objective function is denoted as $g(x|\lambda^i)$. The algorithm employs N ants for solving these single-objective subproblems. Ant i represents the subproblem i. The algorithm has two concepts: 1) neighborhood $B(i)$ (as described for MOEA/D), and 2) groups. The N ants are grouped into K groups by clustering their corresponding weight vectors. The ants in the same group share one pheromone matrix which contains the learned information about their position of the Pareto region. Each group is intended to approximate to a small range of the PF.

[1]CUDA C Programing Guide v5.5. http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf

The algorithm maintains: (I) $\tau^1, ..., \tau^K$, where τ^j is the current pheromone matrix for group j, storing its learned knowledge about the sub-region of *PF* that it aims at approximating; (II) $\eta^1, ..., \eta^N$, where η^i is the heuristic information matrix for subproblem i, which is predetermined before the construction solution starts; (III) *EP*, which is the external archive containing all the non-dominated solutions found so far.

First, the algorithm generate the N initial solutions, the heuristic information matrices, and the pheromone information matrices. Then, at each iteration the MOEA/D-ACO executes the following steps:

1. Generate N solutions according a probabilistic rule;
2. Update *EP*;
3. Update the pheromone matrices according with the new solutions that were constructed by ants in group j and have just been added to *EP*;
4. Check the solutions on the neighborhood and updates the solutions if there is a solution that: 1) is better than its current solutions; 2) has not been used for updating other old solutions. This mechanism makes collaboration among different ant groups (sharing information).
5. The algorithm stops if a criterion is met.

B.2.1 Parallel MOEA/D-ACO for MOKP

Data structure of Pheromone Matrices: Each candidate solution is a 0-1 *n-Dimensional* vector, where n is the number of items. So, the pheromone matrix for group j is $\tau^j = (\tau^j_1, ..., \tau^j_n)$. The approach *Max-Min* [Stützle e Hoos, 2000] is used, so there are boundaries to maximum and the minimum value of τ. All the τ^j_k is initially with $\tau_{max} = 1$ for all $j = 1, ..., K$ and $k = 1, ..., n$.

Heuristic Information matrices: The heuristic information matrix for ant i is $\eta^i = (\eta^i_1, ..., \eta^i_n)$, where η^i_k is to measure the desirability, learned from the domain knowledge before the search. The value of kth in η^i for ant i is

$$\eta^i_k = \frac{\sum_{l=1}^{m} \lambda^i_l p_{l,k}}{\sum_{l=1}^{m} w_{l,k}} \tag{B.1}$$

where $p_{l,k}$ is the profit of item k with the objective l, $w_{l,k}$ is the weight of item k with the objective l. ***EP*** is initialized empty.

Solution Construction: At each subproblem N, the probability of each $item_k$ is denoted as ϕ_k calculated by Equation B.2. The solution vector starts empty, which means that the *n-Dimensional* vector starts with 0 in all dimensions. The items are randomly (using roulette wheel selection) added to the solution one by one, respecting the constraints. The probability is calculated as follows:

$$\phi_k = \frac{(\tau^i_k + \Delta \times x^i_k)^\alpha \times (\eta^i_k)^\beta}{\sum_{k \in I} \phi_k} \tag{B.2}$$

where I is the feasible items; τ^i_k is the pheromone of item k in the subproblem i; $\Delta \times x^i_k$ is a private knowledge; η^i_k is the heuristic value of item k in the subproblem i; α, β and Δ are control parameters. After all ants have constructed their solutions the *EP* is updated with the new solutions. Solutions that are dominated by the new solutions are removed.

Update the Pheromone Matrices: Let Π be the set of all the new solutions that satisfy: (a) they were constructed by the ants in group j of the current iteration; (b) they were just added to EP and (c) in which $item_k = 1$ in *n-Dimensional* vector . Then, τ_k^j, i.e., the pheromone trail value of item k for group j, is updated as follows:

$$\tau_k^j := \rho\tau_k^j + \sum_{x\in\Pi} \frac{1}{\sum_{l=1}^{m}\sum_{k=1}^{n} p_{lk} - g(x|\lambda^j)} \tag{B.3}$$

where ρ is the persistence rate of the old pheromone trails; p_{lk} is the profit of item k on the knapsack l and $g(x|\lambda^j)$ is the objective function to subproblem λ^j. According to [Ke et al., 2013], in this update scheme, pheromone matrix τ^j stores some statistical information of good solutions found so far for the task of group j.

B.2.2 Parallel MOEA/D-ACO for MOTSP

Data structure of Pheromone matrices: Each group j has pheromone trail $\tau_{k,l}^j$ for a link between two different cities k and l. The approach *Max-Min* is also used. All the $\tau_{k,l}^k$ is initialized to $\tau_{max} = 1$.

Heuristic Information matrices: Each ant i has an heuristic information value $\eta_{k,l}^i$ for a link between cities k and l. The value of heuristic information are initialized as

$$\eta_{k,l}^i = \frac{1}{\sum_{j=1}^{m} \lambda_j^i c_{k,l}^j} \tag{B.4}$$

where m is the number of objectives, $c_{k,l}^j$ is the cost between k and l with respect the objective j.

Solution Construction: Assume that ant i is in group j, and its full-scale current solution $x^i = (x_1^i, ..., x_n^i)$. Ant i constructs its new solution following the steps:

1) First, the probability of choosing a link is set. For $k, l = 1, ..., n$ set

$$\phi_{k,l} = [\tau_{k,l}^i \times In(x^i, (k,l)]^\alpha (\eta_{k,l}^i)^\beta \tag{B.5}$$

where α and β are control parameters. ϕ represents the attractiveness of the link between cities k and l to ant i. The indicator function $In(x^i, (k,l))$ is equal to 0 if link *(k,l)* is already in tour x^i or 1 otherwise.

2) Ant i first randomly selects a city to start the tour. After, each city is chosen using the roulette wheel selection. Suppose that its current position is k and it has not completed its tour. It is chosen city l to visit from C (cities not visited so far), according to the following probability by the roulette wheel selection:

$$\frac{\phi_{k,l}}{\sum_{s\in C} \phi_{s,l}} \tag{B.6}$$

3) If the ant has visited all the cities, return its tour.

Pheromone Update: Let Π be the set of all the new solutions that satisfy: (a) were constructed by the ants in group j in the current iteration; (b) were just added to EP; (c) contain the link between cities k and l.

The pheromone trail value of link *(k,l)* for group j, is updated as follows:

$$\tau_{k,l}^j := \rho\tau_{k,l}^j + \sum_{x\in\Pi} \frac{1}{g(x|\lambda^i)} \tag{B.7}$$

where ρ is the persistence rate of the old pheromone trail. As mentioned, τ_{max} and τ_{min} are used to limit the range of the pheromone.

B.3 Parallel Implementation of MOEA/D-ACO

This section briefly describes the CUDA architecture and then reports the decisions that we made to implement a parallel version of MOEA/D-ACO.

CUDA allows developers to run blocks of code, known as *kernels*, directly on the GPU using a parallel programming interface and using familiar programming languages. CUDA parallel programming model has a hierarchy of thread groups called *grid, thread blocks* and *threads*. When a *kernel* function is invoked, it is executed *N* times in parallel by *N* different *CUDA threads*. A single grid is organized by multiple blocks, each of which has equal number of threads.

CUDA threads may access data from multiple memory spaces during their execution. All threads have access to the same *global memory*, which is implemented as an off-chip DRAM of the GPU, and has large capacity, say, 1.5-6 Gigabytes, but its access latency is very long. Each thread block has *shared memory* visible to all threads into a block and with the same lifetime as the thread block. The *shared memory* is an extremely fast on-chip memory with lower capacity, say, 16-48 Kbytes. Each thread has its local (private) memory on-chip, called *register memory*. Registers are the fastest form of storage and each thread within a block has access to a set of fast local registers that are placed on-chip. Each thread can only access its own registers; moreover the number of registers is limited per block, so blocks with many threads will have fewer registers per thread. The efficient usage of the memory types is a key for CUDA developers to accelerate applications using GPU. When threads accesses to continuous locations in a row of a 2-dimensional array *(horizontal access)*, the continuous locations in address space of the global memory are accessed in the same time *(coalesced access)*. From the structure of the global memory, the coalesced access maximizes the bandwidth of memory access. On the other hand, the stride access *(vertical access)* needs a lot of clock cycles [Uchida et al., 2012].

Cecilia et al., 2013 noted that the existing task-based approach of mapping one ant per thread is not suited to the GPU, because each thread must store each ant's memory (data structures of a solution) and this approach works only for small solutions but quickly becomes problematic with larger solutions, as there is limited shared memory and registers available for each block. Based on this issue, the studies [Dawson e Stewart, 2013, Delevacq et al., 2013, Cecilia et al., 2013] adopted a novel data parallel approach that maps each ant to a block. All threads within the thread block then work in cooperation to perform a common task such as tour construction.

The performance on GPU can be drastically affected by the use of a costly math function like *powf()* see Eq.(B.2). Fortunately, there are analogous CUDA functions which map directly to the hardware level (like *__powf()*), although this comes at the expense of some loss of accuracy. [Cecilia et al., 2013] shown a comparison using *__powf()* on an equation which improve the execution time.

First, the algorithm was implemented sequentially and the parallel implementation was made based on it. We base our parallelization strategy on the works [Dawson e Stewart, 2013, Delevacq et al., 2013, Cecilia et al., 2013] and adopt a data-parallel approach mapping each ant to a thread block. The parallel implementation consists of three CUDA parts (initialization, tour construction and pheromone update) and one CPU part (*EP* updated). The *EP* is updated in host (sequential) because each new solution needs to be analyzed at time to update the *EP*. In parallel,

it would be hard to guarantee correctness even using atomic functions. The details of the three GPU parts for the MOKP and MOTSP are described as follows.

B.3.1 Parallel approach for the MOKP

Initialization: The algorithm allocates memory and the relevant data structures: *N-Dimensional* ants vector (which contains the partial solution, the accumulated profit and its group), *n-Dimensional* items vector (which contains the profit and weight of each item) and the *m-Dimensional* knapsack vector (which contains the capacity of each knapsack).

The ants need to choose items in probabilistic way using the roulette wheel selection. The algorithm initializes the random seeds using the CURAND[2], which is a library that provides a pseudorandom number generator on the GPU by NVIDIA.

Solution Construction: Each subproblem i is associated with a thread block. So the N ants (subproblems) construct their solution in parallel. To improve the performance, the number of access to global memory is decreased, allocating some structures on shared memory and register memory. So, the data structure of an ant (solution) is saved on its respective thread register memory; the items vector is also copied to each thread register memory because they are accessed many times; and the data structure of m knapsacks are placed on the shared memory because they are accessed few times. When an ant ends its solution construction the ant is copied to the *N-Dimensional* vector allocated on the global memory. In [Delevacq et al., 2013], the authors report that using the shared memory and registers for these structures would restrict the algorithm to limited number of ants and this restriction would grow linearly with the problem size. The *N-Dimensional* ants vector is copied to host to update the *EP*.

Pheromone Update: A new *kernel* is invoked, now each block corresponds to a group and each thread corresponds to an ant that satisfies Π (see Section B.2.1). The data structure of items is copied to device again, and placed in shared memory. The first step of the update is the pheromone evaporation, which is trivial to parallelize as all pheromone matrices are evaporated by a constant factor ρ. In the second step, each group updates its pheromone matrix in parallel. For each ant in group j that was added to *EP* and in which $item_k = 1$ in *n-Dimensional* vector updates the pheromone trail τ_k^j following Equation B.3, so we do not need to use atomic operations to guarantee correctness.

B.3.2 Parallel approach for the MOTSP

Initialization: Give n cities, the city-to-city distances are loaded in an $(n \times n)$ matrix for each objective (recall, $d_{k,l} = d_{l,k}$). The N ants are loaded to store each ant current tour and tour length. A kernel is invoked to calculate the distance between the cities and set the pheromone matrix initialization. The heuristic value and probability to be chosen is executed during the tour construction, so we do not need to allocate a vector with size N (number of subproblems) to store the N heuristic values for each link *(k,l)*. Also, the algorithm initializes the random seeds using the CURAND[3].

Tour Construction: Each subproblem i is associated with a thread block. So the N ants (subproblems) construct their tour in parallel. The weigh vectors are allocated on shared memory because they are accessed few times. Each ant stores its structure (current tour, tour length, cities not visited, and its group) on the local memory (registers) of a thread. The matrix of cities $(n \times n)$ is stored on the global memory, however, the treads access only continuous location

[2]CUDA Toolkit Guide v4.1 CURAND

[3]CUDA Toolkit Guide v4.1 CURAND

in a row of the matrix $(n \times n)$, i.e., when the current city of a tour is k, the threads access only the k row of the matrix *(k,l)*, where $l = (1,...,n)$. As we mentioned the coalesced access to the global memory is a key issue to accelerate the computation. When an ant ends its tour construction the ant is copied to the vector N-dimensional allocated on the global memory. When all N ants end their tour construction, the *N-Dimensional* vector of ants is copied back to host to update *EP*.

Pheromone Update: A new kernel is called to update the pheromone. Now each thread block corresponds to a group and each thread corresponds to an ant that satisfies Π (see Section B.2.2). The matrix $(n \times n)$ is allocated on the global memory. The pheromone update uses the same approach that the MOKP.

B.4 Experiments

Our sequential algorithm achieves similar results to original MOEA/D-ACO [Ke et al., 2013] in terms of solution quality. In this section we have attained the comparison between the implementations (sequential and parallel) in terms of solution quality and execution time. We have used twelve instances from [Zitzler e Thiele, 1999] for the MOKP, and nine different combinations of instances for the MOTSP from Paquete repository[4] (with a large number of cities compared with studies [Mora et al., 2011, Nebro e Durillo, 2010a]).

We have used the *Weighted Sum* as the scalarizing function. The quality of the solutions is evaluated by means of *Hypervolume* indicator. To evaluate the execution time, we show the speedup of the parallel implementation against the sequential counterpart.

The implementations were made using an NVIDIA GeForce GTX680 that contains 1536 CUDA cores and has a processor speed of 1058 MHz. It uses 32 threads per warp and up to 1024 threads per thread block with maximum shared memory size of 64 Kb. The CPU is an Intel i7-380QM and has 4 cores with support 8 threads with a clock speed of 3.60 GHz. The implementation was written and compiled using the CUDA toolkit 5.0 for C with the Nsight Eclipse environment executed under Ubuntu 12.04.

All the statistics are based on 30 independents runs.

B.4.1 Results from MOKP

The ACO parameters setting was the same used in [Ke et al., 2013]: $\alpha = 1, \beta = 10, \rho = 0.95$. The algorithms stop after 300 generations. The test instance with 750 items and 4 objectives was infeasible due to registers constraints.

Solution Quality: Also based on [Ke et al., 2013], we set the number of subproblems to *N=300* and number of groups to *K=10*. The average and standard deviation of hypervolume for the 30 runs are summarized in Table B.1. The name of the test instances are abbreviated, for example, the instance with 500 items and 2 knapsacks is called 500-2. The highest hypervolume value for each instance is highlighted in bold face. The Wilcoxon statistic test [Derrac et al., 2011] was applied and showed that the results do not have significant difference at 0.99 level of confidence, i.e., the results show that to parallelize the MOEA/D-ACO does not decrease the solutions quality.

Execution time: The parameters number of subproblems and the number of groups are set to $N = 300$ and $K = 10$ respectively and the execution time is analyzed on the different test instances. Table B.2 presents the average execution time (in seconds) for eleven test instances and the speed up of the parallel against the sequential. The results show a speedup up to 19x faster than the sequential implementation.

[4]Available at: http://eden.dei.uc.pt/ paquete/tsp/

Table B.1: Average hypervolume and standard deviation obtained

Instance	Sequential		Parallel	
	Average	Stand. Dev.	Average	Stand. Dev.
100-2	**6.711E+05**	6.213E+02	6.709E+05	3.643E+02
250-2	5.651E+06	4.127E+03	5.651E+06	5.972E+03
500-2	**1.476E+07**	1.002E+04	1.475E+07	7.016E+03
750-2	**3.915E+07**	4.521E+04	3.908E+07	4.123E+04
100-3	**9.240E+08**	1.256E+06	9.232E+08	1.484E+06
250-3	1.199E+10	1.050E+07	**1.200E+10**	8.038E+06
500-3	7.381E+10	1.249E+08	**7.401E+10**	1.656E+08
750-3	**1.896E+11**	2.295E+08	1.881E+11	2.564E+08
100-4	**8.089E+11**	4.409E+09	8.058E+11	2.618E+09
250-4	1.986E+15	1.601E+13	**2.013E+15**	1.725E+13
500-4	**6.173E+16**	7.504E+14	6.081E+16	4.521E+14

Table B.2: Average execution times in seconds (s)

Instance	Sequential (CPU)	Parallel (GPU)	Speedup x
100-2	62.92	5.69	11.065
250-2	431.68	27.75	15.55
500-2	1952.95	105.74	18.47
750-2	4574.43	233.68	19.58
100-3	76.58	9.53	8.03
250-3	447.85	38.49	11.63
500-3	1692.39	123.62	13.69
750-3	3582.08	245.44	16.6
100-4	96.47	23.65	4.08
250-4	368.725	39.13	9.42
500-4	1306.98	137.43	9.51

As reported in Table B.2, the number of items affects the execution time. When the number of items increases (see test instances: 100-2, 250-2, 500-2, 750-2), the speedup increases, because the number of items affects the solution construction phase which is executed in the parallel stage. Thus, when the algorithm consumes more time in the parallel stage the speedup increases.

In addition, as shown in Table B.2, the number of objectives also affects the execution time. When the number of objectives increases (see test instances: 500-2, 500-3, 500-4), the speedup decreases because with more objectives more solutions become non-dominated with each other on the objective space, consequently, the size of *EP* increases at each generation consuming more time of the algorithm in the sequential part. For example, with 500-2 test instance, one generation found 58 non-dominated solutions (19% of the solutions) with final size of $|EP|$=190; with 500-4 test instance, one generation found 255 non-dominated solutions (85% of the solutions) with the final size higher than 4000 solutions. The number of non-dominated solutions is an issue on many-objective optimization problem.

The number of subproblems also affects the speedup. Figure 1 shows the speedup with $N = \{150, 300, 450\}$ on the 500-2 test instance. Each subproblem corresponds to a thread block in the parallel solution construction phase, so higher the number of subproblems higher the number of thread blocks will be executing in parallel which increases the speedup. However,

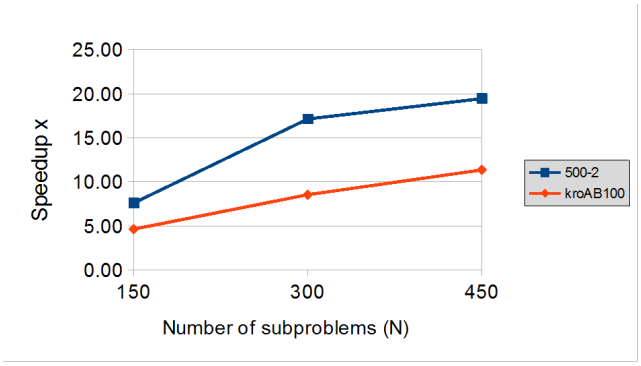


Figure B.1: Speedup of parallel vs. sequential algorithm on 500-2 (MOKP) and kroAB100 (MOTSP) test instances with number of subproblems $N = \{150, 300, 450\}$. The number of groups is fixed *K=10* to 500-2 and *K=3* to kroAB100.

Table B.3: Average hypervolume and standard deviation obtained

Instance	Sequential		Parallel	
	Average	Stand. Dev.	Average	Stand. Dev.
kroAB100	1.860E+10	2.439E+07	**1.861E+10**	1.938E+07
euclidAB300	1.654E+11	4.820E+07	**1.654E+11**	6.680E+07
euclidAB500	**5.162E+11**	7.569E+07	5.162E+11	6.367E+07
kroABC100	**2.618E+15**	1.616E+12	2.596E+15	2.488E+12
euclidABC300	5.461E+16	4.415E+13	**5.446E+16**	4.830E+13
euclidABC500	**3.148E+17**	2.802E+14	3.129E+17	2.213E+14
kroABCD100	2.156E+20	1.229E+19	**2.551E+20**	5.244E+17
euclidABCD300	1.307E+22	1.218E+20	**1.337E+22**	9.224E+19

because of the limited size of CUDA memories (shared memory and registers) a higher number of subproblems can be infeasible.

B.4.2 Results from MOTSP

The ACO parameters setting was the same used in [Ke et al., 2013]: $\alpha = 1, \beta = 2, \rho = 0.95$. The algorithms stop after 1000 generations. The test instance with 500 cities and 4 objectives was infeasible due to CUDA memory constraints.

Solution Quality: We set the number of subproblems and number of groups to *N=300* and *K=3* respectively. The average and standard deviation of hypervolume for the 30 independent runs are summarized in Table B.3. The highest hypervolume value for each instance is highlighted in bold face. The Wilcoxon statistic test was applied and showed that the results do not have significant difference at 0.99 level of confidence as the same in MOKP.

Execution time: Table B.4 presents the average execution time (in seconds) for eight test instances and the speed up of the parallel against the sequential. The results with 300 subproblems and 3 groups show a speedup up to 8x faster than the sequential counterpart.

As showed in Table B.4 the number of objectives affects the execution time for the same reason on the MOKP. When the number of objectives increases (see test instances: *kroAB100*, *kroABC100* and *kroABCD100*) the speedup decreases.

As on the MOKP, the number of subproblems affects the speedup. Figure 1 shows the speedup with $N = \{150, 300, 450\}$ on the kroAB100 test instance. With 450 subproblems the

Table B.4: Average execution times in seconds (s)

Instance	**Sequential (CPU)**	**Parallel (GPU)**	**Speedup x**
kroAB100	172.02	20.14	8.54
euclidAB300	1327.92	169.10	7.85
euclidAB500	2870.92	459.00	6.25
kroABC100	491.46	186.18	2.64
euclidABC300	1617.44	474.19	3.41
euclidABC500	3726.14	798.40	4.67
kroABCD100	909.85	430.93	2.11
euclidABCD300	1895.88	582.07	3.26

parallel algorithm achieves a speedup up to 11x faster than the sequential counterpart. However, because of the limited size of CUDA memories a higher number of subproblems can be infeasible.

B.5 Final considerations

In this chapter, we have proposed a parallel implementation of the MOEA/D-ACO on GPU using CUDA, where the both *solution construction* and *pheromone update* stages are executed on GPU. Our results show a speedup up to 19x faster for the MOKP and 11x faster for the MOTSP using a reasonable number of subproblems and groups. The algorithm achieves a higher speedup for the MOKP because the algorithm makes fewer accesses to global memory when compared with the MOTSP. On the MOTSP the data structure for the $n \times n$ matrix of cities was infeasible to allocate on the shared memory or registers.

Other aspects which impacts on the execution time: (1) A great number of objectives degrades the speedup because it affects the size of the *EP*, and consequently its update stage which is executed on CPU; (2) A great number of subproblems increases the performance because each subproblem executes its solution construction in parallel, but a high number can be an issue due to the memory limits.

Some aspects to consider for future work: (1) How to deal with the memory limits for largest test instances and a higher number of subproblems; (2) Improving the execution time using other parallelization approaches, e.g., a parallel version of the roulette wheel selection as in [Cecilia et al., 2013] (which has been not implemented here due the architecture of the GPU available); (3) And also, a parallel approach for the *EP* update mechanism.