Multi-Swarm Algorithms in Dynamic Software Project Scheduling Problem

Mikael T. Cerqueira de Jesus[1], **Leila Maciel**[1], **André Britto**[1]

[1]Departamento de Computação – Universidade Federal de Sergipe (UFS)

mikael.jesus@dcomp.ufs.br, andre@dcomp.ufs.br

Abstract. *The Dynamic Software Project Scheduling Problem (DSPSP) is a Multiobjective Optimization Problem that still not solved in Search Based Software Engineering Area. To address this problem many algorithms have been used and some of them achieved good results. Among them, we can highlight Multiple Population Multiobjective Particle Swarm Optimization algorithm, called Multi-Swarms. The objective of this work is to explore dynamic optimization techniques applied to a Multi-Swarm algorithm to solve DSPSP. To do so, new multi population strategies are proposed. The proposed approaches are confronted to a basic Multi-Swarm algorithm over a DSPSP instance, through the analysis of quality indicators and statistical tests.*

Resumo. *O Problema de Escalonamento Dinâmico de Projetos de Software (DSPSP) é um problema ainda em aberto no ramo da Engenharia de Software Baseada em Busca. Na literatura, vários algoritmos foram utilizados e alguns deles se mostraram promissores. Entre estes estão os algoritmos de Otimização por Enxame de Partículas com Múltiplas Populações chamados de Multi-Swarm. O objetivo desse trabalho é explorar técnicas de otimização dinâmica para resolver o DSPSP. Para isso, novas estratégias baseadas em múltiplas populações são propostas. As abordagens propostas são comparadas com um algoritmo Multi-Swarm, utilizando-se uma instância do DSPSP e fazendo-se uma análise com indicadores de qualidade e testes estatísticos.*

1. Introdução

Problemas de Engenharia de Software são geralmente complexos, pois lidam com a necessidade de satisfação de restrições e com informação ambígua e imprecisa, de modo que não possuem abordagem exata para serem resolvidos e que podem possuir uma infinidade de boas soluções [Colanzi et al. 2013]. Essas características resultam em uma complexidade de decisão muito alta e tornam uma resolução manual algo inviável e custoso em relação a tempo. Há uma área de pesquisa da engenharia de software chamada Engenharia de Software Baseada em Busca (SBSE, do inglês *Seach Based Software Engineering*), que mapeia esses problemas em problemas de otimização e propõe aplicar técnicas de busca para resolvê-los[Harman et al. 2012].

Um dos problemas de Engenharia de Software é o Problema de Escalonamento de Projetos de Software (SPSP, do inglês *Software Project Scheduling Problem*), formulado em [Alba and Chicano 2007]. Esse problema é caracterizado como um problema de alocação que consiste em designar empregados às tarefas de um projeto, onde é necessário achar a alocação que ao mesmo tempo minimize o custo e a duração do projeto. Isso se

torna um problema, pois são objetivos conflitantes. À medida que o custo com o projeto diminui, sua duração tende a aumentar e vice-versa. Neste caso, existe mais de um objetivo, conflitantes entre si, a ser otimizado simultaneamente, sendo assim um problema de otimização multiobjetivo [Coello et al. 2007].

Apesar de existirem formulações e possíveis soluções para problemas como o SPSP, na vida real as coisas não acontecem da mesma forma. As formulações citadas até agora simulam problemas em ambientes estáticos, que permanecem inalterados do início ao fim. Porém, no mundo real, na grande maioria das vezes ocorrem imprevistos os quais alteram a formulação do projeto e podendo tornar inválida uma solução encontrada no modelo estático. Essa abordagem pode ser considerada uma extensão do SPSP, chamada Problema de Escalonamento Dinâmico de Projetos de Software (DSPSP, do inglês *Dynamic Software Project Scheduling Problem*). Nessa versão do problema, o objetivo é alocar empregados às tarefas considerando os eventos e incertezas que podem ocorrer no projeto ao longo do tempo, como por exemplo a saída de empregados, a chegada de tarefas urgentes ou a reestimativa dos esforços das tarefas. Esses eventos podem introduzir mudanças nas variáveis de decisão do problema e mudança no cálculo das funções objetivo. Assim, ele é definido como um Problema de Otimização Multi-Objetivo Dinâmico [Helbig and Engelbrecht 2013].

Nesse contexto, em [Shen et al. 2016] foi proposta uma modelagem para o DSPSP. Nessa formulação são considerados mais 2 objetivos a serem otimizados, totalizando assim 4 objetivos. Categorizando-o assim como um problema de otimização com muitos objetivos ou *Many-Objective Problem*(MaOP) [Ishibuchi et al. 2008].

Essa modelagem teve uma implementação proposta em [do Amaral 2018], que implementou o modelo no *framework spsp-jmetal*. O *framework* desenvolvido é baseado na versão 5.2 do *framework jmetal* proposto em [Durillo and Nebro 2011]. O modelo implementado permite a utilização de diferentes meta-heurísticas e técnicas de otimização dinâmica para a resolução do problema.

Entre essas possíveis meta-heurísticas, o framework disponibiliza um algoritmo baseado na Otimização por Enxame de Partículas Multiobjetivo (MOPSO, do inglês *Multiobjective Particle Swarm Optimization*) com múltiplas populações, o MS2MO (*Multi-Swarm Multi-Strategy Algorithm for Many-Objective Optimization*), algoritmo também proposto em [do Amaral 2018]. Esse algoritmo baseia-se em disparar vários enxames de PSO, de forma que se comuniquem periodicamente entre si, baseados na topologia definida, trocando informações sobre o estado das suas soluções. O MS2MO foi confrontado com os algoritmos NSGA-II e SMPSO em [do Amaral 2018], obtendo bons resultados.

Porém, apesar de explorar uma estratégia de múltiplas populações, no MS2MO, todas sua populações se comportam de forma semelhante, tendo uma mesma forma de inicialização da população durante a ocorrência dos eventos dinâmicos. Dentre as estratégias da otimização dinâmica, múltiplas populações podem ser usadas para explorar os espaço de busca de diferentes maneiras, fazendo assim que o algoritmo esteja mais apto se adaptar aos eventos dinâmicos que podem ocorrer.

Assim, este trabalho tem como objetivo explorar técnicas de otimização dinâmica para resolver o DSPSP. Para isso, novas estratégias baseadas em múltiplas populações são propostas e são desenvolvidas extensões do algoritmo MS2MO. Essas abordagens são

baseadas em diferentes formas de reinício das populações quando um evento dinâmico ocorre.

Este trabalho busca identificar se as estratégias propostas melhoram ou não o resultado do algoritmo MS2MO. As abordagens propostas são confrontadas com um algoritmo Multi-Swarm básico em uma instância do DSPSP. Na comparação dos resultados, é usado o indicador de qualidade Hipervolume e é usado o teste estatístico de Wilcoxon para medir a significância.

O reste deste artigo está organizado da seguinte maneira: a Seção 2 define o problema DSPSP. Na seção 3 apresenta o algoritmo MS2MO básico e as extensões propostas. Os experimentos são discutidos na seção 4 e, por fim, as conclusões são apresentadas na seção 5.

2. Problema de Escalonamento Dinâmico de Projetos de Software

2.1. Problema de Escalonamento de Projetos de Software

O SPSP contém essencialmente 2 conjuntos de elementos, o conjunto dos empregados e o conjunto das tarefas. Todo empregado possui seu conjunto de habilidades, tempo de dedicação máximo e um salário associado. Por exemplo, um determinado empregado pode possuir a habilidade de Programação e Banco de Dados, uma dedicação máxima de 75% do seu dia e salário mensal de R$ 2.500,00, enquanto outro pode ser também especialista em programação e web design, poder dedicar 80% do seu dia e receber um salário de R$ 3.000,00. A dedicação máxima é definida como a razão entre as horas do dia dedicadas ao projeto e a duração do expediente do empregado.

Os empregados são denotados por e_i , o salário dos mesmos por e_i^{salary}, o conjunto de habilidades por e_i^{skills}, a dedicação máxima por e_i^{maxded}, e o estado que mostra se o empregado está ou não disponível no momento t denotado por $e_i^{available(t)}$, onde i varia de 1 até $|E|$ (número de empregados presentes no projeto).

No projeto existe o conjunto **SK** que simboliza o conjunto de todas as habilidades dos empregados. Cada tarefa possui no máximo 2 pré-requisitos: Pré-requisitos de habilidades necessárias e de outras Tarefas, ou seja, para determinada tarefa ser iniciada, uma outra deve ser completada. Para determinada tarefa ser completada é necessária uma quantidade de esforço total expressa em pessoa/mês. O conjunto de habilidades necessárias para determinada tarefa ser realizada é denotada por T_j^{skills}, o esforço associado por T_i^{effort}, e o estado que mostra se a tarefa j já foi finalizada em um momento t é denotado por $T_j^{unfinished(t)}$, onde i varia de 1 até $|T|$ (número de tarefas). A precedência de tarefas e seus pré-requisitos é modelado em um grafo orientado e acíclico, o Grafo de Precedência de Tarefas (TPG, do *inglês Task Precedence Graph*), onde cada vértice do grafo representa uma tarefa e cada aresta (u,v) representa a relação de pré-requisito de u para v.

Além dos elementos apresentados existem também restrições que precisam ser satisfeitas, elas são:

- Todas as habilidades necessárias para executar as tarefas devem estar presentes no **SK**.
- Nenhum empregado pode exceder sua dedicação máxima.

- A quantidade de empregados não pode exceder um limite chamado *Maximum headcount constraints*.

No *SPSP* são considerados dois objetivos, o custo e a duração do projeto. As fórmulas para o cálculo de ambos a partir da matriz de dedicação estão presentes na seção 2 em [Alba and Chicano 2007] e a formulação matemática das restrições na seção 3.6 em [Shen et al. 2016].

A solução do problema é dada em forma de uma matriz de dedicação X_{ExT}, tal que $X_{i,j}$ representa a dedicação do empregado i para a tarefa j. Por exemplo, na Tabela 1 o empregado e_1 utiliza 80% de sua dedicação mensal, ou seja, de sua carga horária, para trabalhar na tarefa t_4. O objetivo é achar a solução X_{ExT} que minimize o custo e a duração do projeto. A Tabela 1 representa um exemplo de matriz de dedicação de um projeto.

Tabela 1. Exemplo de Solução do SPSP

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_6
e_1	0,00	0,00	0,00	0,80	0,40	0,20	0,00	0,50
e_2	1,00	0,80	0,75	0,25	0,80	0,20	0,00	0,35
e_3	0,50	0,60	0,00	0,00	1,00	0,00	1,00	0,40
e_4	0,75	0,45	0,00	0,60	0,30	0,75	0,00	0,50

Fonte: [do Amaral 2018]

2.2. Formalização do Problema Dinâmico

Para a abordagem dinâmica são adicionados 2 objetivos, robustez e estabilidade. A robustez observa e classifica o comportamento de determinada solução em ambientes incertos, penalizando as soluções que menos se adaptam a ambientes alternativos derivados do ambiente original. O valor dela representa o quanto a avaliação da solução varia nos ambientes alternativos.

A estabilidade é um objetivo que busca ser minimizado e somente entra em questão em um segundo momento, onde ocorreu uma mudança de ambiente. Esse objetivo calcula o quanto as soluções encontradas no momento atual diferem das soluções encontradas no escalonamento passado. Portanto, pretende-se achar a melhor solução para o novo problema que mais se pareça com a solução anterior, por diversos motivos, principalmente de engajamento de funcionários. As fórmulas para o cálculo de ambos os objetivos, estabilidade e robustez, estão presentes em [Shen et al. 2016].

Para a simulação de possíveis situações do mundo real são introduzidos eventos dinâmicos, eles são: Empregado sai, Empregado volta e Nova tarefa.

Para melhorar a convergência do algoritmo proposto, [Shen et al. 2016] definiram três heurísticas para a inicialização das populações no momento de reescalonamento. Essas heurísticas são: Ajuste proativo, uso de características históricas, inserção aleatória. Essas heurísticas são a base das extensões propostas nesse artigo.

A técnica de ajuste proativo consiste em explorar as características peculiares de cada evento dinâmico, para assim tentar ajustar as soluções do escalonamento anterior. A ideia é que essas soluções ajustadas componham a população inicial do enxame no escalonamento atual. O tratamento peculiar para cada evento são descritos abaixo:

- **Empregado sai**: Se um evento de saída de empregado é disparado todas as tarefas para as quais ele não está escalado continuam intactas, mas a saída do mesmo pode gerar duas situações diferentes. Se após a saída dele o restante do time escalado para a tarefa preencher os requisitos de habilidades necessários para a tarefa, tudo continua o mesmo. Caso contrário outros empregados disponíveis juntam-se ao time para satisfazer a restrição. Caso isso não seja possível, não é possível determinar solução factível para o problema.
- **Empregado volta**: Se um evento de retorno de empregado é disparado e há tarefas nas quais as habilidades dele possam ajudar, então ele é escalado para tais de modo a acelerar o término da mesma. Do contrário, o escalonamento continua o mesmo.
- **Nova tarefa**: Caso apareça uma nova tarefa, o escalonamento dos empregados disponíveis emrelação às demais tarefas continua o mesmo, e o escalonamento para a nova tarefa é definido aleatoriamente.

A técnica de uso de características históricas consiste em compor uma parcela da população inicial com soluções de escalonamentos passados. Essa técnica vem da premissa de utilizar soluções de escalonamentos onde o ambiente de projeto era parecido com o atual. A técnica de inserção de partículas aleatórias busca inserir partículas aleatórias nos exames, buscando manter a diversidade das soluções.

3. Múltiplos Enxames Aplicados ao DSPSP

Nessa sessão são apresentados o algoritmo MS2MO proposto em [do Amaral 2018], e as modificações no *framework* propostas pelo trabalho presente.

3.1. Algoritmo MS2MO

O MS2MO é um algoritmo que usa a técnica de múltiplos enxames de partículas para a resolução de MaOP's por meio da comunicação periódica entre os enxames.

No MS2MO, cada enxame possui sua própria população. Durante o processo de busca, esses enxames se comunicam entre si através de um arquivo externo, que é atualizado periodicamente com as melhores soluções de cada enxame. Cada um deles também possui um arquivo interno que somente ele pode alterar. Esse arquivo interno se trata das melhores soluções encontradas pela respectiva população. No momento da troca de informações, as soluções são trocadas diretamente entre os enxames. Essas trocas ocorrem de acordo com a topologia definida. Ao receberem as soluções, os enxames não sobrescrevem seu arquivo interno, apenas tentam inserir as novas soluções nele, baseados no tipo de arquivamento utilizado [do Amaral 2018].

O algoritmo utiliza para cada enxame um MOPSO. No modelo implementado o algoritmo utiliza instâncias do algoritmo SMPSO(do inglês, *Speed-constrained Multi-objective* PSO) [Nebro et al. 2009], adaptadas para atender as características da otimização dinâmica. A principal modificação foi poder especificar por parâmetro um conjunto de soluções para compor a população inicial do enxame, que torna possível a utilização de diferentes heurísticas dinâmicas na resolução do problema.

O algoritmo usa a topologia de anel para os enxames. Essa topologia define cada enxame possuindo apenas 2 enxames vizinhos, com os quais trocam informações diretamente. Entretanto, todos estão conectados indiretamente, assim, há uma influência mútua

entre todos os enxames. O algoritmo recebe por parâmetro o número de avaliações da função objetivo, número de enxames, número de partículas por enxame e a periodização da comunicação entre os enxames. Para a aplicação do mesmo ao DSPSP, o modelo implementado simula um ambiente real de projeto. Esse modelo, portanto, requer como parâmetro características da instância de projeto utilizada. Características essas que incluem: quantidade de empregados e suas habilidade; quantidade de tarefas e suas respectivas habilidades requeridas; eventos dinâmicos que acontecerão ao decorrer do projeto [do Amaral 2018]. Além disso, receberá como parâmetro as porcentagens de uso de cada uma das heurísticas dinâmicas descritas na seção 2.2.

Algoritmo 1: Pseudocódigo da modificação realizada

```
Entrada: Número de avaliações da função objetivo; Número de enxames; Número de partículas
         por enxame; Momento de comunicação entre os enxames; Eventos dinâmicos; Uso da
         heurística de aproveitamento histórico para cada enxame; Uso da heurística de ajuste
         proativo para cada enxame
Saída: Conjunto de soluções não-dominadas entre todos os enxames
início
    Rode o MS2MO para encontrar um cronograma inicial para o projeto;
    para cada evento dinâmico faça
        se for o primeiro reescalonamento então
            Rode o MS2MO;
        fim
        senão
            populacoes ← ∅;
            para Cada enxame s do MS2MO faça
                populacaoReparada ← Conjunto de soluções reparadas baseadas na taxa de
                  ajuste proativo de s;
                populacaoHistorica ← Conjunto de soluções criado baseado na taxa de
                  aproveitamento histórico de s;
                populacao ← populacaoReparada ∪ populacaoHistorica;
                populacoes ← populacoes ∪ populacao;
            fim
            para Cada enxame s do MS2MO e população em populacoes faça
                Monte s utilizando população como população inicial;
            fim
            Rode o MS2MO;
        fim
    fim
fim
```

3.2. Modificações Propostas

A apesar do MS2MO utilizar a técnica de múltiplas populações para resolver o problema, o conceito é pouco explorado no modelo. Uma vez que todos os enxames são configurados para possuírem um comportamento semelhante, visto que possuem a mesma parametrização para o uso das heurísticas dinâmicas e a cada escalonamento todos os enxames se baseiam na mesma população inicial para inicializar suas respectivas populações iniciais. Uma forma diferente de explorar a técnica seria fazer com que cada enxame possua um comportamento diferente em relação aos outros.

A proposta apresentada nesse trabalho é justamente utilizar diferentes parametrizações para os enxames e fazer que a cada escalonamento cada enxame inicie

independentemente no espaço de busca. Essa ideia tem a premissa de tornar o início das busca no reescalonamento menos determinístico. Uma vez que cada enxame se comporte de forma diferente, cada um deles podem encontrar soluções melhores anteriormente não exploradas. Como há uma influencia mútua entre todos os enxames, achar essa solução influenciaria na busca de todos os outros. Graças a isso, além de cada enxame utilizar de uma parametrização diferente, o método proverá combinação entre todas elas no resultado.

Assim, serão analisados os impactos da aplicação de diferentes heurísticas dinâmicas por enxame e da inicialização independente de cada enxame na solução do problema. O Algoritmo 1 resume o funcionamento do modelo modificado.

Para tal modificação, os parâmetros que se referiam as porcentagens de uso de cada uma das heurísticas dinâmicas são agora vetores de valores de tamanho igual ao número de enxames. Seja o vetor referente ao uso do ajuste proativo chamado de *repairedSolutions* e o referente ao uso de informações históricas chamado de *historicalSolutions*. A atribuição de parâmetros se dá de forma que, para todo elemento $i, 1 <= i <= numeroDeExames$, repairedSolutions[i] e historicalSolutions[i] representam os parâmetros de heurísticas dinâmicas do enxame i. O Algoritmo 1 resume o funcionamento do modelo modificado.

4. Resultados e Discussões

Nesse trabalho foi feita a comparação entre o algoritmo modificado, com diferentes parâmetros, e o algoritmo e parâmetros base propostos em [do Amaral 2018]. Na seção 4.1 serão definidas as nomenclaturas utilizadas para os algoritmos e pamatrizações comparadas, as instâncias utilizadas em cada experimento e as métricas utilizadas para a avaliação do desempenho de cada algoritmo. Em seguida, na seção 4.2 são apresentados e discutidos os resultados obtidos em cada experimento.

4.1. Planejamento do Experimento

4.1.1. Algoritmos e Parâmetros

Para uma comparação justa, para todas as execuções dos algoritmos foram fixados os valores de 16 enxames, população de tamanho 200, tamanho dos enxames 100, e 12.000 avaliações de função objetivo. Para o algoritmo modificado foram usadas 5 parametrizações diferentes. Para fins de melhor entendimento cada parametrização será chamada por um nome diferente:

- MS2MODYNAMICMod-RE: Todos os 16 enxames utilizando apenas a estratégia de reparo proativo.
- MS2MODYNAMICMod-H: Todos os 16 enxames utilizando apenas a estratégia de aproveitamento histórico.
- MS2MODYNAMICMod-R: Todos os 16 enxames utilizando apenas inserção aleatória a cada escalonamento.
- MS2MODYNAMICMod-A: Cada um dos 16 enxames utilizando uma configuração heterogênea diferente de estratégias.
- MS2MODYNAMICMod-C: Cada um dos 16 enxames utilizando a mesma parametrização base.

De modo que assim, possa ser avaliado o impacto de cada estratégia heurística dinâmica e da inicialização independente nos resultados. A tabela 2 lista a parametrização do Algoritmo MS2MODYNAMICMod-A.

Tabela 2. Parametrização do algoritmo MS2MODYNAMICMod-A

Enxames	Estratégias(%)		
	Aleatórias	Histórico	Ajuste Proativo
Enxame 1	50	20	30
Enxame 2	0	0	0
Enxame 3	0	25	75
Enxame 4	0	50	50
Enxame 5	0	75	25
Enxame 6	0	100	0
Enxame 7	25	0	75
Enxame 8	25	25	50
Enxame 9	25	50	25
Enxame 10	25	75	0
Enxame 11	50	0	50
Enxame 12	50	25	25
Enxame 13	50	50	0
Enxame 14	75	0	25
Enxame 15	75	25	0
Enxame 16	100	0	0

Fonte: Próprio autor

O algoritmo base de [do Amaral 2018] será chamado de MS2MODYNAMIC. Nele 30% das partículas de todas as populações utilizam a estratégia de ajuste proativo, 20% usa a estratégia de aproveitamento histórico e 50% utiliza inserção aleatória.

Para discutir com melhor precisão os resultados, para cada algoritmo foram realizadas 20 execuções. Visto que nem todas as execuções possuem o mesmo número de escalonamentos, foram considerados os dados relativos aos primeiros 135 escalonamentos, de modo que todas execuções possuíram no mínimo esse valor.

4.1.2. Instâncias

Os experimentos foram conduzidos sobre uma das instâncias artificiais de simulação de projeto criadas e utilizadas em [Shen et al. 2016] e em [do Amaral 2018].

As instâncias são escritas no formato sT*N*_dT*M*_E*X*_sK*Y*-*Z*, de forma que *N* é o número de tarefas iniciais do projeto, *M* é o número de tarefas que serão introduzidas no projeto ao decorrer do tempo, *X* é o número de empregados, *Y* é o número mínimo de habilidades requeridas para cada tarefa e *Z* é o máximo. Entre elas, 20% dos empregados trabalham em tempo parcial, com dedicação aleatória que varia no intervalo [0.5, 1], outros 20% podem trabalhar horas extras, com dedicações variando no intervalo [1.0, 1.5] e os 60% restante trabalham com dedicação 1.0. As instâncias utilizadas em todos os experimentos com os algoritmos testados foram as sT10_dT10_E15_sK4-5, sT10_dT10_E5_sK4-5 e sT20_dT10_E15_sK4-5. Para fins de simplicidade, as mesmas serão chamadas de instância 1, instância 2 e instância 3, respectivamente.

4.1.3. Métricas

A métrica de desempenho utilizada é o Hipervolume, métrica normalmente utilizada na avaliação de problemas de otimização multiobjetivo. O cálculo do Hipervolume de uma fronteira de Pareto consiste em definir um ponto de referência no espaço de busca, de modo que esse ponto deve representar uma solução que não domina nenhuma outra presente na fronteira em questão. O Hipervolume se trata do volume entre a fronteira e esse ponto de referência, como o ponto de referência representa a pior solução no conjunto, quanto mais distante do ponto de referência melhor a fronteira, logo, quanto maior o Hipervolume melhor a fronteira.

O método para geração dos pontos de referência foi o mesmo usado em [do Amaral 2018]. O programa, a cada escalonamento, gera um arquivo com os objetivos da fronteira encontrada. Esses objetivos então são normalizados, considerando os resultados de todos os algoritmos comparados. Com a normalização, os valores possíveis dos objetivos se encontram no intervalo [0.0,1.0]. Assim, o ponto de referência utilizado no primeiro escalonamento (onde a estabilidade não é considerada) é (1.1,1.1,1.1), e nos demais escalonamentos é (1.1,1.1,1.1,1.1).

4.1.4. Testes Estatísticos

A comparação dos resultados obtidos foram feitos através dos testes estatísticos não paramétricos de Wilcoxom considerando um nível de significância de 0,05, para a comparação de duas amostras, e de Friedman, para comparação de 3 ou mais amostras de modo simultâneo [Derrac et al. 2011]. Foram também comparadas as médias dos hipervolumes de cada ponto de escalonamento dos algoritmos, além das médias e desvios padrão gerais.

Para a análises dos resultados foram usandos scripts em Python, com o uso das bibliotecas: SciPy, Matplotlib e scikit-posthocs.

4.2. Resultados

Nessa seção são apresentadas as comparações dos resultados obtidos a partir dos experimentos executados. Na tabela 3 são apresentadas as médias e desvios padrões gerais dos hipervolumes obtidos em todos os algoritmos para cada instância.

4.2.1. Impacto da inicialização independente dos enxames

Para analisar isoladamente o impacto da inicialização independente dos enxames, foram comparados os resultados do algoritmo base com os do MS2MODYNAMICMod-C. A figura 1 ilustra a comparação da média dos hipervolumes a cada de reescalonamento de ambos os algoritmos. Ao analisar a curva de resultados observa-se que, apesar dos dois algoritmos manterem a mesma parametrização para todos os enxames, o MS2MODYNAMICMod-C apresentou resultados inferiores na maioria dos escalonamentos. Sugerindo que a inicialização independente dos enxames não surte efeito positivo no algoritmo. Nos resultados das instâncias 1 e 3 essa conclusão fica mais clara. Na

instância 2 percebe-se que o algoritmo converge mais rapidamente e quase se aproxima ao algoritmo base, porém apresenta diversas quedas de hipervolume ao decorrer do projeto.

O teste de Wilcoxom foi aplicado a série de resultados obtidos por ambos os algoritmos em todas as instâncias. O p-value obtido foi de $1.433436E-22$, $2.602088E-23$, $7.341722E-24$, indicando uma clara diferença estatística entre os dois algoritmos.

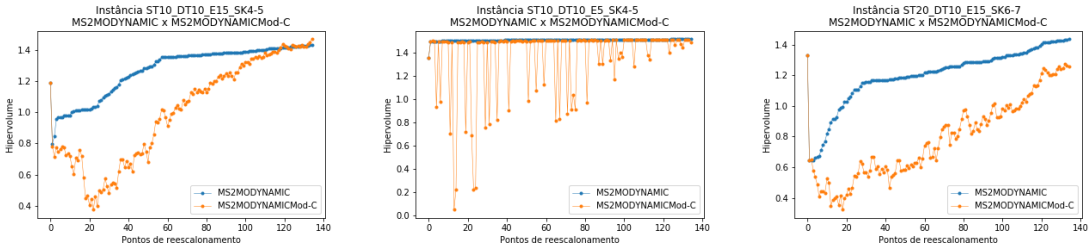

Figura 1. Comparação dos hipervolumes a cada ponto de escalonamento entre os algoritmo MS2MODYNAMIC e MS2MODYNAMICMod-C

4.2.2. Impacto do uso de diferentes heurísticas dinâmicas por enxame

Para analisar o impacto do uso de diferentes heurísticas dinâmicas por enxame, foram comparados os resultados obtidos entre os algoritmos MS2MODYNAMICMod-A, MS2MODYNAMICMod-H, MS2MODYNAMICMod-R, MS2MODYNAMICMod-RE, MS2MODYNAMICMod-C. A figura 2 mostra a comparação das médias dos hipervolumes obtidos a cada ponto de escalonamento entre todos os algoritmos em todas as instâncias do problema.

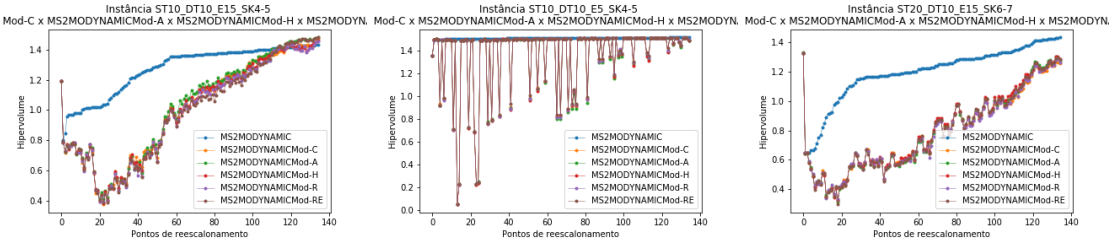

Figura 2. Comparação das médias dos hipervolumes a cada ponto de escalonamento em todas as instâncias

Ao analisar os gráficos nota-se que, no geral, o MS2MODYNAMIC obteve melhores resultados que o restante dos algoritmos que utilizavam das modificações propostas. Apresentando resultados mais consistentes ao decorrer de todo o projeto e se mostrando inferior apenas em escalonamentos mais avançados.

Nas instâncias 1 e 3, observa-se que a curva dos resultados do algoritmo proposto apresenta um comportamento semelhante em todas as suas versões. Os resultados decaem até por volta de um mesmo ponto de reescalonamento, e ao decorrer das execuções a curva torna-se quase crescente, podendo atingir valores melhores do que o algoritmo base. Na instância 2, percebe-se o mesmo comportamento do MS2MODYNAMICMod-C no restante dos algoritmos, uma convergência melhor que nas outra instâncias, porém ainda com diversas quedas de hipervolume.

Tabela 3. Médias e desvios padrão gerais dos valores de hipervolume obtidos dos algoritmos.

Algoritmos	Instância 1	Instância 2	Instância 3
MS2MODYNAMIC	**1.420(0.156)**	**1.51(0.044)**	**1.390(0.199)**
MS2MODYNAMICMod-A	0.966(0.323)	1.38(0.290)	0.890(0.337)
MS2MODYNAMICMod-C	0.955(0.317)	1.38(0.289)	0.891(0.332)
MS2MODYNAMICMod-H	0.953(0.321)	1.38(0.288)	0.894(0.335)
MS2MODYNAMICMod-R	0.944(0.317)	1.38(0.288)	0.883(0.327)
MS2MODYNAMICMod-RE	0.944(0.318)	1.38(0.288)	0.885(0.335)

Fonte: Próprio autor.

No geral, entre as versões do algoritmo proposto, percebe-se que houveram resultados muito semelhantes em cada ponto de escalonamento em cada uma das instâncias do problema. A figura 3 mostra um *heatmap* que indica o grau de significância das diferenças entre todos os algoritmos dois a dois considerando o *p-value*. Ao observar a figura pode-se afirmar que, para um grau de significância de 0,05, não há diferença estatística entre as diferentes versões do algoritmo. Logo, pode-se afirmar que o uso de diferentes heurísticas dinâmicas por enxame não influi impacto no algoritmo utilizando a inicialização independente dos enxames.

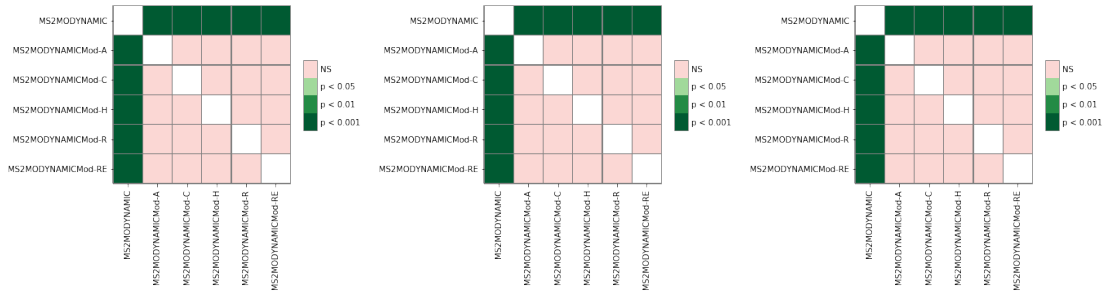

Figura 3. Comparação pareada das diferenças estatísticas entre os algoritmos em todas as instâncias

5. Conclusões

Esse trabalho teve como objetivo explorar técnicas de otimização dinâmica aplicadas à estratégia de multiplas populações. Para isso, foi utilizado o modelo implementado por [do Amaral 2018] para a simulação de ambientes reais de projeto e o algoritmo MS2MO proposto pelo mesmo. No modelo, foram feitas modificações que exploram o viés multi-populacional do algoritmo. Essas modificações basearam em dois conceitos chave: A utilização de diferentes configurações de uso de heurísticas dinâmicas em cada enxame e a inicialização independente deles no espaço de busca a cada escalonamento.

Com a análise dos dados, notou-se que o algoritmo base no geral mostrou-se mais eficiente que todas as intâncias do algoritmo proposto. Essas últimas que apresentaram resultados bastante semelhantes entre si, apesar de terem parametrizações totalmente diferentes. Sugerindo que o uso das diferentes heurísticas dinâmicas não surte muito efeito no algoritmo modificado.

Foi observado que as curvas dos hipervolumes de todas as intâncias do algoritmo proposto decaem sempre até por volta de um mesmo ponto de escalonamento. Ao decorrer dos escalonamentos a curva se torna quase crescente e em escalonamentos mais

avançados chegam até a apresentar valores de hipervolume melhores que o algoritmo base. Isso sugere que o algoritmo pode atingir bons resultados a longo prazo.

Dessa forma, surge a oportunidade de investigar o comportamento da curva apresentada pelas diferentes versões do algoritmo proposto. Apresentar uma queda de desempenho nos escalonamentos iniciais e melhores resultados ao decorrer das interações. Além disso, abre-se a oportunidade de explorar os diferentes tipos de comunicação entre os enxames para melhorar a convergência do algoritmo.

Agradecimentos

Este estudo foi financiado em parte pela CAPES - Código de financiamento 001 e pelo CNPq, projeto Universal número 425861/2016-3.

Referências

Alba, E. and Chicano, J. F. (2007). Software project management with gas. *Information Sciences*, 177(11):2380 – 2401.

Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*, volume 2. Springer.

Colanzi, T. E., Vergilio, S. R., Assunção, W. K. G., and Pozo, A. (2013). Search based software engineering: Review and analysis of the field in brazil. *Journal of Systems and Software*, 86(4):970 – 984. SI : Software Engineering in Brazil: Retrospective and Prospective Views.

Derrac, J., García, S., Molina, D., and 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.

do Amaral, R. O. M. (2018). Otimização com muitos objetivos por múltiplos enxames aplicada ao escalonamento dinâmico de projetos de software. Master's thesis, Universidade Federal de Sergipe.

Durillo, J. J. and Nebro, A. J. (2011). jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760 – 771.

Harman, M., Mansouri, S. A., and Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)*, 45(1):11.

Helbig, M. and Engelbrecht, A. P. (2013). Performance measures for dynamic multi-objective optimisation algorithms. *Information Sciences*, 250:61 – 81.

Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2008). Evolutionary many-objective optimization. In *2008 3rd International Workshop on Genetic and Evolving Systems*, pages 47–52.

Nebro, A. J., Durillo, J. J., Garcia-Nieto, J., Coello Coello, C. A., Luna, F., and Alba, E. (2009). Smpso: A new pso-based metaheuristic for multi-objective optimization. In *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making(MCDM)*, pages 66–73.

Shen, X., Minku, L. L., Bahsoon, R., and Yao, X. (2016). Dynamic software project scheduling through a proactive-rescheduling method. *IEEE Transactions on Software Engineering*, 42(7):658–686.