Distribuição de Tarefas em uma Fábrica de Software: Equilibrando entregas e lucro em otimização Mono-Objetivo

Arthur Gomes Faria
Gabriel Lopes Machado
Jonas Reis Jr
Lucas da Silva

Engenharia de Sistemas - Escola de Engenharia
Universidade Federal de Minas Gerais - UFMG
Belo Horizonte - Brasil

Palavras chaves: Otimização Multiobjetivo, Scrum, Pesquisa Operacional, Problema das Multiplas Mochilas, Modelagem Matemática

1. Resumo

No dia a dia, há diversos problemas em que é preciso tomar decisões em relação a recursos, sejam eles financeiros, de tempo ou esforços que visam aumentar nossa produtividade ou economia em qualquer que seja o escopo do nosso trabalho. Para tal pode-se recorrer a modelagens matemáticas que visam otimizar um problema procurando a melhor solução ainda que não ótima mas que chegue no resultado que buscamos. Essas modelagens serão exploradas para encontrar uma solução mais adequada para o problema elencado.

2. Introdução

Durante o desenvolvimento e manutenção de um produto de software, muitas são as tarefas elencadas para sua completude. Essas tarefas normalmente incluem, por exemplo, construção de servidores, implementação de lógicas, desenhos de interface, validação com o cliente, codificação de testes, entre várias outras. Para adicionar mais complexidade, cada sistema trabalha com suas próprias tecnologias que devem ser conhecidas pela equipe que o constrói e o mantém.

Uma das formas de se construir um produto de software na atualidade é terceirizando o trabalho e contratando uma "fábrica de software", empresa especializada na construção de tais produtos. Normalmente, essas empresas são divididas em uma série de equipes (também chamadas de Squads) [2] com um conhecimento específico de uma tecnologia e uma capacidade limitada de execução. Por exemplo, uma equipe de desenvolvedores de X tecnologia consegue entregar Y tarefas dentro de um período de tempo T. Caso a fábrica adote a metodologia SCRUM de desenvolvimento, temos que Y pode ser também chamado de Story Points e T de Sprint (usualmente um período de duas semanas) [1].

Ademais, dentro de uma fábrica de software, é fundamental haver um equilíbrio entre a satisfação do cliente e a satisfação interna, seja entregando uma grande quantidade de tarefas para o cliente como também escolhendo as que trazem maior lucro para a empresa. Dessa forma, é importante modelar o trabalho a ser feito para que os dois objetivos possam ser vislumbrados, mantendo o balanço saudável do lucro e das entregas.

O objetivo então consiste em encontrar uma modelagem matemática para uma distribuição de tarefas entre as Squads respeitando as restrições de capacidade, conhecimento e disponibilidade e garantir que todos os clientes sejam atendidos gerando um bom lucro para a empresa.

Este problema se assemelha à modelagem da distribuição de itens em mochilas [4, 3] com uma série de restrições e adicionando a complexidade da necessidade de se entregar tarefas para todos os clientes, mesmo que o valor pago por ele seja inferior aos outros.

3. Modelagem do Problema

Para começar a modelagem do problema, é importante definir o espaço das variáveis e suas representações. O problema proposto foi inicialmente modelado de forma semântica e depois transformado com o devido rigor matemático e metodologico nas seguites variáveis:

- M: Quantidade de Squads ou Equipes.
- i: Indice da Squad.

- N: Quantidade de clientes.
- j: Indice do cliente.
- P_j: Quantidade de tarefas requeridas do cliente j.
- k: Indice da tarefa.
- cs_i: Capacidade maxíma de Sotry Points da Squad i.
- ct_{jk}: Quantidade de Story Points que a tarefa k do cliente j consome.
- es_i: Codifica a especialidade da squad i.
- et_{jk}: Codifica a especialidade que a tarefa k do cliente j requer.
- c_{jk}: Lucro de realizar a tarefa k do cliente j.
- u_{ijk}: Condição boleana se indica se o Squad i realizará a tarefa k do cliente j.

A partir desta lista, é possível então derivar as equações matemáticas de otimização, assim como as restrições. Iniciando pela primeira: maximização (do lucro), tem-se:

$$\textbf{maximize: } f_1 = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{P_j} c_{jk} \cdot u_{ijk}$$

Como também é importante atender a todos os clientes, tem-se a modelagem da satisfação entregue pela fabrica de software aos seus clientes:

$$\textbf{maximize: } f2 = \sum_{j=1}^{N}\frac{1}{P_j}\sum_{k=1}^{P_j}\sum_{i=1}^{M} u_{ijk}$$

Essa função objetivo acima é uma simplificação grosseira da ponderação da satifação de cada cliente. Porém, muito importante para tornar o problema mais simples.

Também relacionadas às funções objetivas, tem-se as restrições do problema:

sujeito a:

$$1)\sum_{j=1}^{N}\sum_{k=1}^{P_j} ct_{jk} \cdot u_{ijk} \leq cs_i, \forall i = 1, ..., M$$

$$2)(es_i - et_{jk}) \cdot u_{ijk} = 0, \forall i = 1, .., M; j = 1, .., N; k = 1, .., P_j$$

$$3)\sum_{i=1}^{M}\sum_{k=1}^{P_j} u_{ijk} > 0, \forall j = 1, ..., N$$

$$4)\sum_{i=1}^{M} u_{ijk} \leq 1, \forall j = 1, ..., Nek = 1, ..., P_j$$

$$5)\mathbf{u} \in \mathbb{S}^{M(P_1+...+P_N)}$$

Onde:

1. Garante que toda Squad não vai ultrapassar o seu limite de tarefas;
2. Garante que uma squad só vai receber tarefas que abrangem sua própria especialidade;
3. Garante que todo cliente terá pelo menos uma tarefa realizada pela fabrica de software;
4. Garante que no máximo uma squad vai alocar a tarefa k da empresa j;
5. Define a natureza binária das variáveis de decisão e o total de variáveis.

4. Algoritmo Mono-Objetivo

Para execução do algoritmo Mono-objetivo foi usado a biblioteca do Gurobi Optimizer no python, a gurobipy, usada comumente para solução de problemas de otimização linear, programação linear inteira mista e quadrática. Para a solução de problemas continuos ou modelos inteiros mistos o optimizador usa o algoritmo barrier ou simplex no entanto para problemas mais complexos ele usa uma heurística para selecionar o melhor algoritmo para solução ou otimização simutânea com o uso diversos algoritmos simultâneamente e retorna a solução do primeiro no final. Para execução do projeto foi usado uma amostra de 5 Squads e 10 clientes com os seguintes dados:

- P_j: 100 , 30 , 5 , 130 , 55 , 40 , 15 , 30 , 50 , 10
- cs_i: 400 , 100 , 600 , 1000 , 150
- ct_{jk}: 2 , 10 , 50 , 3 , 5 , 6 , 15 , 30 , 18 , 30
- es_i: 1 , 1 , 1 , 1 , 1
- et_{jk}: 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
- c_{jk}: 2 , 10 , 50 , 3 , 5 , 6 , 15 , 30 , 18 , 30

5. Resultados

Sendo f_1 a função que visa maximizar o lucro da empresa e f_2 a busca de maior satisfação o cliente o algoritmo computacional usado gerou os seguintes resultados:
Otimizando para f_1

- Quantidade de tarefas realizadas: 359
- Função de satisfação: 6.6842307692307665
- Lucro obtido: 12430.0

Otimizando para f_2

- Quantidade de tarefas realizadas: 388
- Função de satisfação: 8.07333333333333
- Lucro obtido: 11780.0
- Solução utópica: f_1 - 12430; f_2 - 8.0733333
- Solução antitópica: f_1 - 11780; f_2 - 6.6842

6. Conclusão

Podemos observar que a satisfação do cliente em f_1 é menor comparado com f_2 o que ocorre com o inverso quando se analisa o lucro. Tal situação antagônita pode ser melhorada usando a Fronteira Pareto em um problema Multi-Objetivo, o ótimo de Pareto traz um equilíbrio onde os recursos alocados estão distríbuidos na melhor forma possível e que qualquer rearranjo diferente dele prejudicaria mais um objetivo em detrimento do outro.

Referências

[1] Sutherland J. adn Schwaber K. The scrum guide, 2020.

[2] House D. Squads: o modelo de organização que vem tomando conta das startups., 2018.

[3] Jonas Krause, Rafael S Parpinelli, and Heitor S Lopes. Proposta de um algoritmo inspirado em evolução diferencial aplicado ao problema multidimensional da mochila. *Anais do IX Encontro Nacional de Inteligência Artificial–ENIA. SBC, Curitiba*, 2012.

[4] João Carlos Heringer Moreira. Uma abordagem com multi-mochilas multidimensionais para o problema de alocação de ações de redução de perdas na distribuição de energia. Master's thesis, UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO, Vitória, 2015.