6
Conclusões e trabalhos futuros

Nesta dissertação foi proposto um *framework* para facilitar a implementação de heurísticas baseadas em construção de vocabulário. O desenvolvimento foi fundamentado em extensa revisão bibliográfica sobre a técnica e em boas práticas de engenharia de software, como *frameworks* orientados a objetos e padrões de projeto. O projeto do *framework* foi pautado pelas premissas de baixo acoplamento com outros métodos, facilidade na variação das implementações e flexibilidade na representação de dados.

São fornecidas implementações dos operadores clássicos de extração e combinação de palavras definidos por Glover, explicados na Seção 2.1 e formalizados na Seção 3.5. Esses operadores já demonstraram sua eficácia na resolução em outros problemas [1, 8].

O operador clássico para extração de palavras INT foi disponibilizado nas duas implementações. Uma obtém palavras a partir do máximo possível de interseções de soluções cujo tamanho da palavra gerada seja maior do que um dado valor mínimo. A outra abordagem busca palavras mais extensas, e por isso com uma menor quantidade de soluções utilizadas respeitando um valor mínimo de soluções utilizadas. A primeira combina mais soluções para obter palavras mais consistentes e a segunda combina menos soluções para obter palavras mais extensas.

As implementações dos operadores clássicos utilizam vetores de inteiros como tipo genérico para representação de soluções. Para facilitar o acoplamento da heurística gerada pelo *framework* a métodos já existentes, foi utilizado o padrão de projeto *adapter*. Sendo assim, possibilita-se utilizar os operadores clássicos em problemas com representação específica. Para isso, basta que a solução na sua representação original seja traduzida para sua representação como vetor de inteiros.

Durante o desenvolvimento do *framework*, foi verificado que é necessário avaliar as soluções candidatas ao repositório de boas soluções para evitar o armazenamento de soluções muito parecidas, piorando o desempenho do método. Optamos pela diversidade das soluções no repositório e, por isso, foi implementada a avaliação pela distância de Hamming para comparar duas

soluções. Assim, permite-se garantir que as soluções no repositório sejam diferentes em pelo menos um número mínimo de posições do vetor de inteiros.

Como um estudo de caso, foram geradas três aplicações a partir do *framework* para a resolução do problema de seqüenciamento da produção de carros. Em cada aplicação foi obtida uma heurística diferente pela utilização de diferentes configurações dos pontos flexíveis: representação de solução, métodos para extrair e combinar palavras e gerenciamento do repositório de soluções.

O desenvolvimento dessas aplicações foi realizado de modo incremental, avaliando se acrescentar construção de vocabulário com os operadores clássicos à estratégia proposta por Ribeiro et al. [26, 27, 29], apresentada na Seção 4.2, melhoraria na resolução do problema de seqüenciamento de carros. Para representar uma solução do problema foi adotada a representação por vetor de inteiros (`VetInt`). Como a solução do problema nessa estratégia é representada por uma estrutura complexa, foi necessário converter essas soluções em objetos `VetInt`, tarefa que foi bem implementada no *framework*.

Os operadores clássicos, por serem operadores genéricos, podem ser utilizados na resolução de outros problemas. Entretanto, os resultados computacionais demonstraram que com a utilização desses operadores não foi possível melhorar a resolução do problema de seqüenciamento de carros.

Para esse problema, o operador clássico de combinação de palavras não conseguiu formar frases. Para contornar essa limitação, foi considerada a possibilidade de utilizar-se outras formas de representação da solução. Algumas formas alternativas são a representação por vetor de adjacência, como definido em [8], ou a representação por vetor com tipo de carros, onde considera-se carros de mesma cor e restrições como iguais. Entretanto, a formação de ciclos na recuperação da solução inviabilizou na prática a utilização dessas abordagens para esse problema.

Pelo fato da composição dos operadores no *framework* ter sido definida utilizando o padrão de projeto *strategy*, é possível variá-los em tempo de execução, permitindo assim, a comparação entre implementações numa mesma execução. Essa característica foi explorada nos experimentos computacionais quando diversos parâmetros de métodos foram avaliados para uma mesma instância.

Dentre os trabalhos futuros ficam a implementação de outros operadores de extração e combinação de palavras como os que são os definidos em [25]. Esses operadores são baseados em técnicas de mineração de dados. Soluções são representadas por conjuntos de itens. Padrões são definidos como subconjuntos de itens que ocorrem num número significativo de soluções. O processo de obter esses padrões é o bem conhecido problema em mineração de dados chamado

Mineração de Conjunto Freqüente de Itens. São utilizadas estratégias distintas pela variação de como se gera e utiliza o conjunto de padrões. Por exemplo, uma é chamada maior padrão híbrido. Ela utiliza apenas o maior padrão encontrado num conjunto de padrões. Esse conjunto é formado pelos padrões presentes em pelo menos um percentual de elementos do conjunto de soluções elite. Outra estratégia, chamada maior suporte híbrido, procura pelo padrão mais freqüente no conjunto suporte dado um tamanho mínimo. É parecido com a forma de utilizar o operador clássico INT, mas feito de uma forma mais sofisticada.

Outro trabalho futuro é utilizar a distância de Levenshtein [20], também conhecida como distância de edição, para avaliar soluções candidatas ao repositório de soluções. A distância de Hamming é definida somente para vetores de mesmo comprimento. Dados dois vetores, a distância de Hamming é o número total de valores que são diferentes em posições respectivas dos vetores. Na distância de Levenshtein, a distância entre dois vetores é dada pelo número mínimo de operações necessárias para transformar um vetor no outro. São consideradas as operações de inserção, remoção ou substituição de um valor. Assim, pode-se determinar quão semelhante são dois vetores, mesmo de tamanhos diferentes. Essa métrica é muito utilizada em corretores ortográficos. Como a distância de Levenshtein é mais sofisticada e necessita de mais processamento, sua utilização pode degradar o desempenho da heurística que gastará mais tempo para avaliar as soluções candidatas ao repositório.

Dentre as vantagens em se utilizar o *framework* proposto neste trabalho, considera-se a reutilização dos componentes desenvolvidos como a melhor. Pois, o desenvolvimento de heurísticas para outros problemas se torna mais rápido e fácil aproveitando-se esses componentes. Esse benefício é ampliado com a utilização do *framework* em outras situações. Pode-se aproveitar os componentes previamente implementados, bem como aproveitar futuramente os componentes desenvolvidos para outras aplicações. A desvantagem em se utilizar um *framework* é a necessidade de aprendê-lo, no sentido de conhecer seus componentes e a interação entre eles. Como o *framework* proposto neste trabalho possui poucas classes, é fácil aprender sua arquitetura e desenvolver novos componentes para utilizá-lo na resolução de outros problemas. Enfim, o *framework* desenvolvido está apto a ser utilizado na geração de heurísticas para resolução de outros problemas.