# UMA ABORDAGEM DE MATHEURÍSTICA PARA O PROBLEMA BIOBJETIVO DE ROTEAMENTO EM ARCOS CAPACITADOS

**Igor Machado Coelho[1], Daniel Porumbel[2], El-Ghazali Talbi[3]**
**Luiz Satoru Ochi[1] e Luidi Simonetti[1]**

[1] Universidade Federal Fluminense, Niterói, RJ, 24210-240, Brasil

[2]Université d'Artois, Artois, 62000, França

[3]INRIA Lille Nord Europe, Villeneuve d'Ascq, 59650, França

imcoelho@ic.uff.br, daniel.porumbel@univ-artois.fr, el-ghazali.talbi@inria.fr,

satoru@ic.uff.br e luidi@ic.uff.br

***Abstract.*** *This work presents a novel approach inspired in the power of matheuristics (heuristics integrated with exact approaches) to deal with a bi-objective arc routing problem (CARP). The bi-objective CARP is defined over a set of weighted arcs that must be serviced by vehicles of limited capacity and some arcs that does not need to be visited. The objectives are to minimize the longest tour and also the sum of all traversed arcs. This $\mathcal{NP}$-Hard problem has many practical applications such as garbage collection, snow cleaning and inspection on electric wires. We propose an integration between a classic evolutionary multi-objective algorithm, NSGA-II, and an exact decoder based on dynamic programming to evaluate incomplete solution based on the permutation of arcs. Preliminary results show that the proposed approach is well-suited for the CARP and it is extensible to other problems with indirect representation.*

***KEYWORDS****: Bi-Objective Arc Routing Problem, Indirect Representation, Matheuristic, NSGA-II, Multi-Objective Optimization.*

***Resumo.*** *Este trabalho apresenta uma nova abordagem inspirada no poder das matheurísticas (heurísticas integradas a abordagens exatas) aplicada ao problema biobjetivo de roteamento em arcos capacitados (capacited arc routing problem – CARP). O CARP é definido por um conjunto de arcos ponderados que devem ser servidos por veículos de capacidade limitada e alguns arcos que não precisam ser visitados. O CARP biobjetivo busca minimizar a soma de todos arcos visitados e também a rota mais longa. Este problema $\mathcal{NP}$-Difícil tem várias aplicações práticas como coleta de lixo, limpeza de neve e inspeção de redes elétricas. É proposta uma integração entre um algoritmo evolutivo multiobjetivo clássico, o NSGA-II, e um decodificador exato baseado em programação dinâmica para avaliar soluções incompletas baseadas em permutações de arcos. Resultados preliminares mostram que a abordagem é adequada para o CARP e é extensível a outros problemas com representação indireta.*

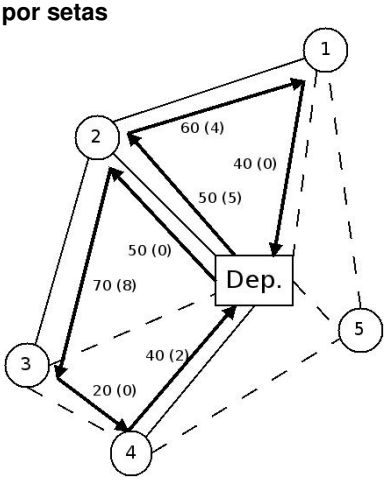
***PALAVRAS-CHAVE****: Problema Biobjetivo de Roteamento em Arcos Capacitados, Representação Indireta, Matheurística, NSGA-II, Otimização Multiobjetiva.*

## 1 Introdução

Este trabalho aborda uma versão biobjetiva do Problema de Roteamento em Arcos (*Capacited Arc Routing Problem* – CARP), que consiste de um grafo com arcos ponderados que devem ser servidos por veículos de capacidade limitada e também arcos sem demanda por serviço. Embora a definição clássica do problema considere arcos na ligação entre nós, na literatura e no presente trabalho são consideradas arestas, sem perda de generalidade. Para formar as rotas, os veículos partem de um depósito comum e servem arestas com demanda positiva, denominados *arestas requeridas*. As arestas requeridas podem ser atravessadas múltiplas vezes, sendo a demanda satisfeita em alguma das visitas e não havendo serviço nas outras, apenas travessia. Travessias sem serviço, denominadas *deadheading*, podem ocorrer em arestas requeridas e também em arestas sem demanda (denominadas *arestas não-requeridas*). A versão biobjetiva do CARP consiste em: minimizar a soma de todas as arestas percorridas; minimizar a rota mais longa percorrida.

A Figura 1 apresenta uma solução com dois veículos de capacidade 10 para um problema com 10 arestas: quatro requiridas e seis não-requiridas. A primeira rota parte do depósito, atravessa uma aresta até o nó 2, depois até o nó 1, depois retorna ao depósito, percorrendo uma distância de 50+60+40=150, com uma carga de 9. A segunda rota parte do deposito, atravessa novamente a aresta que conecta o nó 2 (fazendo *deadheading*), depois vai até o nó 3, depois ao nó 4 e retorna ao depósito com uma carga de 10, percorrendo uma distância de 50+70+20+40=160. Neste exemplo, os dois valores da função objetivo são 310 (soma de distâncias) e 160 (maior rota) para o primeiro e segundo objetivos, respectivamente.

**Figura 1. Exemplo do CARP com um depósito central (retângulo), cinco vértices (círculos), quatro arestas requiridas (traços contínuos), seis arestas não-requeridas (traços pontilhados) e duas rotas marcadas por setas**




Este problema pertence à classe $\mathcal{NP}$-Difícil e tem várias aplicações práticas descritas na literatura como: leitura de medidores (Stern e Dror, 1979), logística para alimentação de gado (Dror *et al.*, 2000), coleta de lixo (Lacomme *et al.*, 2006) e inspeção de trilhos de trem (Lannez *et al.*, 2010). Na versão biobjetiva do CARP introduzida por Lacomme *et al.* (2006), a rota mais longa é também minimizada de forma a respeitar outras restrições práticas, por exemplo, controlar o tempo máximo de trabalho para os motoristas dos veículos. Devido à alta complexidade do problema, abordagens heurísticas geralmente são utilizadas para encontrar soluções de alta qualidade (porém não exatas) para aplicações industriais de

alta escala.

Neste trabalho é apresentado um nova matheurística (heurísticas com abordagens exatas) que combina o poder de um decodificador exato de programação dinâmica com uma metaheurística evolucionária clássica para problemas multi-objetivo: a Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb *et al.*, 2002). O problema é modelado por meio de representação indireta baseada em permutações e um decodificador é responsável por gerar um conjunto de soluções completas para o problema com base em uma permutação de entrada. Esta abordagem estende os arcabouços clássicos multiobjetivos, logo o modelo apresentado para o CARP pode ser aplicado a um grande número de problemas de otimização na literatura. Uma comparação com outros algoritmos da literatura é apresentada para demonstrar que a matheurística proposta é competitiva.

O trabalho é organizado da seguinte forma. A Seção 2 apresenta o CARP biobjetivo em detalhes, descreve o processo de decodificação desenvolvido para o CARP biobjetivo e apresenta trabalhos anteriores no assunto. A Seção 3 estende o arcabouço clássico multiobjetivo para uso de representação indireta com múltiplas soluções decodificadas. Na Seção 4 o modelo estendido é aplicado ao NSGA-II, um algoritmo evolucionário multiobjetivo clássico na literatura. Finalmente, a Seção 5 apresenta os resultados dos experimentos computacionais da matheurística desenvolvida e a Seção 6 conclui o trabalho.

## 2 Problema Biobjetivo de Roteamento em Arcos Capacitados

O CARP biobjetivo é definido formalmente por um grafo $G = (V, E)$, onde $V$ é um conjunto de $n$ nós e $E$ um conjunto de $m$ arestas, na versão não-direcionada do problema apresentada pela literatura. Um subconjunto de arestas $E_R$, com cardinalidade $\tau$, representa o conjunto de arcos requeridos, que devem ser servidos por pelo menos um veículo, de uma frota homogênea com capacidade $W$. Um custo de deslocamento $c_{ij}$ e demanda $q_{ij}$ são associados a cada aresta $(i, j)$, havendo demanda zero para arcos não-requeridos. Rotas $R$ devem ser criadas partindo e retornando do depósito, um nó especial $n_0$. Para cada rota, a soma das demandas não deve exceder a capacidade $W$ do veículo. Os dois objetivos são apresentados pelas Equações (1)-(2).

$$minimizar \left( \sum_{r \in R} \sum_{e_{ij} \in r} c_{ij} \right) \tag{1}$$

$$minimizar \left( \max_{r \in R} \sum_{e_{ij} \in r} c_{ij} \right) \tag{2}$$

Algumas heurísticas clássicas foram desenvolvidas para o CARP, como Augmented Merge Golden e Wong (1981) e Path-Scanning Golden *et al.* (1983). Estas heurísticas são utilizadas em trabalhos seguintes como a Busca Tabu desenvolvida por Hertz *et al.* (2000) e o Algoritmo Memético de Lacomme *et al.* (2004). Em Lacomme *et al.* (2006) a metaheurística multiobjetiva NSGA-II é aplicada à versão biobjetiva do CARP, também utilizando heurísticas construtivas clássicas e operadores de busca de local consagrados para roteamento de veículos.

Em termos de representação indireta, existem algoritmos de *splitting* que trabalham com uma representação de rota gigante (Lacomme *et al.*, 2006), na qual não há separação entre as rotas de cada veículo. Neste tipo de abordagem, o algoritmo de *splitting* fica responsável por decidir quais melhores pontos de retorno ao depósito, transformando a rota gigante em um conjunto de rotas menores que respeitem a capacidade dos veículos. Porém, ao aplicar esta técnica diretamente ao CARP, além de decidir a sequência de visita das arestas requeridas, também é necessário decidir a direção do translado nestas arestas, aumentando enormemente o número de rotas gigantes representadas por esta estrutura. Então, desenvolvemos um algoritmo de decodificação que leva em conta apenas a sequência de visita das arestas, apresentado pelo Algoritmo 1.

**Algoritmo 1**: Decodificação via Programação Dinâmica

**Entrada**: $V$: conjunto de nós $\{v_0 \dots v_n\}$; $E_R$: conjunto de arestas requeridas $\{e_1 \dots e_\tau\}$; $c(e_k)$: custo da aresta $e_k \in E_R$; $q(e_k)$: demanda da aresta $e_k \in E_R$; $W$: capacidade do veículo; $s$: permutação $(e_1 \dots e_\tau)$ do conjunto $E_R$

Dada uma aresta $(v_i, v_j) = e_k \in E_R$, sejam $e_k^0$ e $e_k^1$ os nós $v_i$ e $v_j$, respectivamente
$m(i,j) \leftarrow$ caminho mínimo dos nós $i, j \in V$
$E_R^0 \leftarrow E_R \cup \{e_0\}$, onde $e_0$ representa a aresta $(v_0, v_0)$
$E_R^\tau \leftarrow E_R^0 \,/\, \{e_\tau\}$
// PASSO 1: INICIALIZAR $L$, $T$, $D^0$ E $D^1$
$L$: vetor numérico para arestas $E_R^\tau$
$T, D^0, D^1$: vetor numérico para cada aresta $e_k \in E_R^\tau$
**para** $e_k \in E_R^\tau$ **faça**
    $\Delta \leftarrow \max\{\delta : \sum_{i=1}^{\delta} q(e_{k+i}) \leq W\}$, onde $\Delta$ denota o número máximo de arestas consecutivas a serem servidas após $e_k$, respeitando $W$
    $L(e_k) \leftarrow \Delta$
    $T(e_k), D^0(e_k), D^1(e_k) \leftarrow$ vetor de tamanho $\Delta$;
**fim**
// PASSO 2: COMPUTAR $T$, $D^0$ E $D^1$
**para** $e_k \in E_R^\tau$ **faça**
    $D^0(e_k, 1) \leftarrow m(v_0, e_{k+1}^1) + c(e_{k+1})$
    $D^1(e_k, 1) \leftarrow m(v_0, e_{k+1}^0) + c(e_{k+1})$
    $T(e_k, 1) \leftarrow \min\{D^0(e_k, 1) + m(e_{k+1}^0, v_0), D^1(e_k, 1) + m(e_{k+1}^1, v_0)\}$
    **para** $\delta = 2$ *até* $L(e_k)$ **faça**
        $D^0(e_k, \delta) \leftarrow \min\{D^0(e_k, \delta-1) + m(e_{k+\delta-1}^0, e_{k+\delta}^1) + c(e_k),$
                $D^1(e_k, \delta-1) + m(e_{k+\delta-1}^1, e_{k+\delta}^1) + c(e_k)\}$
        $D^1(e_k, \delta) \leftarrow \min\{D^0(e_k, \delta-1) + m(e_{k+\delta-1}^0, e_{k+\delta}^0) + c(e_k),$
                $D^1(e_k, \delta-1) + m(e_{k+\delta-1}^1, e_{k+\delta}^0) + c(e_k)\}$
        $T(e_k, \delta) \leftarrow \min\{D^0(e_k, \delta) + m(e_{k+1}^0, v_0), D^1(e_k, \delta) + m(e_{k+1}^1, v_0)\}$
    **fim**
**fim**
// PASSO 3: ENCONTRA FRENTE DE PARETO BIOBJETIVA COM PROGRAMAÇÃO DINÂMICA
$M, M^{max}$: matrizes $(e, e')$, $e, e' \in E_R^\tau$, preenchidas com valor $\infty$
$P \leftarrow \{\}$
**para** $e_k \in E_R^\tau$ **faça**
    **para** $e_p \in E_R^\tau$ **faça**
        **para** $\delta = 1$ *to* $L(e_k)$ **faça**
            **se** $M(e_k, e_p) + T(e_k, \delta) \leq M(e_{k+\delta}, e_{p+1})$ **então**
                $M(e_{k+\delta}, e_{p+1}) \leftarrow M(e_k, e_p) + T(e_k, \delta) \leq M(e_{k+\delta}$
                $M^{max}(e_{k+\delta}, e_{p+1}) \leftarrow \max\{M^{max}(e_k, e_p), T(e_k, \delta)\}$
                atualizaConjuntoPareto($P$, $M(e_{k+\delta}, e_{p+1})$, $M^{max}(e_{k+\delta}, e_{p+1})$)
            **fim**
        **fim**
    **fim**
**fim**
**retorna** $P$

No Algoritmo 1, dada uma aresta $(i, j) = e_k \in E_R$, as variáveis $L(e_k)$ armazenam o número de arestas que é possível servir após aresta $e_k$ (sujeito à permutação de entrada $s$),

**Algoritmo 2**: NSGA-II

**Entrada**: $P_S^0$: população inicial
1 $N \leftarrow |P_S|$
2 $Q_S^0 \leftarrow$ `GerarFilhos` $(P_S^0)$
3 $t \leftarrow 1$
4 **enquanto** *critério de parada não satisfeito* **faça**
5 $\quad R_S^t \leftarrow P_S^{t-1} \cup Q_S^{t-1}$
6 $\quad F \leftarrow$ `NonDominatedSort` $(R_S^t)$
7 $\quad P_S^t \leftarrow \{\}$
8 $\quad i \leftarrow 1$
9 $\quad$ **enquanto** $|P_S^t| + |F_i| \leq N$ **faça**
10 $\quad\quad$ `CrowdingDistanceAssignment` $(F_i)$
11 $\quad\quad P_S^t \leftarrow P_S^t \cup F_i[1 \ldots (N - |P_S^t|)]$
12 $\quad$ **fim**
13 $\quad Q_S^t \leftarrow$ `GerarFilhos` $(P_S^t)$
14 $\quad t \leftarrow t + 1$
15 **fim**
16 **retorna** $P_S^t$

respeitando a capacidade $W$ dos veículos. As variáveis $D^0(e_k, \lambda)$ armazenam o custo de servir as arestas $(e_1 \ldots e_{k+\lambda})$ e terminar no vértice $i$, enquanto $D^1(e_k, \lambda)$ fazem o mesmo de $D^0$, mas terminam no vértice $j$ (outra extremidade da aresta $e_k$. As variáveis $T(e_k, \lambda)$ armazenam o custo de servir as arestas $(e_1 \ldots e_{k+\lambda})$ e retornar ao depósito, sendo este o custo completo de uma rota. A seguir, as matrizes $M$ e $M^{max}$ são atualizadas sequencialmente pelo algoritmo de programação dinâmica, atualizando também o conjunto de pareto $P$ com soluções cujos objetivos conflitantes pertencem a uma frente de pareto (soluções não-dominadas).

## 3 Adaptação de metaheurísticas multiobjetivas

Metaheurísticas que buscam resolver problemas combinatórios com múltiplos objetivos podem ser classificadas de acordo com seus três grandes componentes (Talbi, 2009): atribuição de *fitness*, estratégia de elitismo e manutenção de diversidade. O NSGA-II Deb *et al*. (2002) tem componentes que efetuam cada uma dessas funções, que podem ser vistas no Algoritmo 2.

De acordo com Talbi (2009), a atribuição de *fitness* tem o objetivo de guiar a busca e prover um meio de comparar as diversas soluções encontradas durante a busca, podendo ser classificado como: escalar, por critérios, por dominância e por indicadores de qualidade. O tipo de atribuição de *fitness* influencia a qualidade da busca e tem melhores resultados dependendo da estrutura do problema abordado. Algoritmos clássicos como NSGA-II (Deb *et al*., 2002) e SPEA (Zitzler e Thiele, 1999) utilizam a dominância como atribuição de *fitness*, enquanto o IBEA (Zitzler e Künzli, 2004) utiliza indicadores de qualidade como guia da busca. No caso do NSGA-II, a estratégia consiste em ordenar as soluções por frentes de dominância, de forma que as soluções da cada frente não são dominadas entre si nem por soluções de frentes posteriores (linha 6 do Algoritmo 2). Exemplos de abordagens escalares são: método de agregação, no qual objetivos são combinados linearmente (Deb *et al*., 2006); programação por metas, ou *goal programming*, quando o tomador de decisão define níveis de aspiração e metas para cada objetivo, assim busca-se minimizar a diferença entre o objetivo e as metas (Charnes e Cooper, 1977); o $\epsilon$-*constrained method*, que otimiza cada objetivo sujeito à fixação dos outros (Haimes *et al*., 1971); entre outros.

As estratégias de elitismo buscam manter os melhores elementos encontrados na busca dentro do conjunto pareto final retornado pelo método de busca. Algoritmos como NSGA-II mantém os melhores indivíduos na população dada a forma de atribuição de *fitness* juntamente com sua etapa de seleção natural (linha 11 do Algoritmo 2), mas algoritmos como SPEA dependem de uma população extra, denominada *archive*, que contém as melhores soluções encontradas, mesmo que essas não façam mais parte da população corrente. Em Talbi (2009) são apresentadas várias estratégias para manutenção e atualização do *archive* que não serão apresentadas aqui visto que o algoritmo desenvolvido não depende de tais estratégias, assim como o NSGA-II também não depende.

Para se desenvolver um bom algoritmo de busca deve-se manter soluções de alta qualidade, mas também garantir certa diversidade entre essas soluções, papel que é desempenhado pelo componente de manutenção de diversidade (Talbi, 2009). No NSGA-II a estratégia de diversificação é denominada *crowding distance* (linha 10 do Algoritmo 2), e ela atua como forma de desempate quando soluções de mesmo *fitness* são confrontadas. Desta forma, busca-se incluir soluções que cubram as mais diversas áreas da frente de pareto, incluindo elementos com diversos valores para cada objetivo e buscando aumentando o conjunto de pareto para haver uma melhor tomada de decisão posteriormente ao processo de otimização.

De forma a incluir o decodificador exato (Algoritmo 1) no arcabouço de metaheurísticas multiobjetivas como o NSGA-II (Algoritmo 2), algumas adaptações importantes são necessárias. Formalmente, definimos $S$ como o espaço das permutações, ou o espaço das soluções incompletas, utilizando uma função de decodificação $\mathcal{D}$ que mapeia cada solução incompleta a um conjunto de soluções decodificadas (ou soluções completas), que pertencem ao espaço de soluções $X$, ou espaço de soluções CARP.

Como todas as operações básicas serão operadas no espaço $S$ das permutações, é necessário que o decodificador seja capaz de gerar soluções ótimas para o problema considerando que todas possíveis soluções em $S$ sejam dadas como entrada. Como apontado por Lacomme *et al*. (2004), esta característica não se aplica a todos problemas combinatórios. Existem problemas de escalonamento em que, de forma a simplificar operações de cruzamento um decodificador é utilizado, mas para algumas instâncias soluções ótimas não podem ser atingidas (French, 1982).

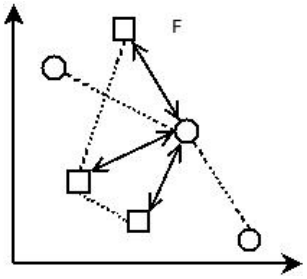
Neste contexto, a atribuição de *fitness* deve ser adaptada para operar com a função de decodificação $\mathcal{D}$ capaz de gerar múltiplos elementos simultaneamente. Dado um problema combinatório $\Pi$ consistindo na minimização dos objetivos (sem perda de generalidade), busca-se um conjunto pareto ótimo (ou conjunto eficiente) $P^*$ consistindo somente de soluções não dominadas $x \in X$, onde $X$ é o espaço das soluções viáveis de $\Pi$. Para cada solução de entrada $s \in S$, a função de decodificação $\mathcal{D}$ gera um conjunto de elementos em $X$. Seja $\mathcal{F}(x \in X) \rightarrow \mathbb{R}$ qualquer técnica para atribuição de *fitness* apresentada na literatura, propõe-se neste trabalho utilizar a *melhor solução* entre todas as decodificadas como representante da permutação de entrada. Desta forma, podemos facilmente aproveitar o procedimento de atribuição de *fitness* do NSGA-II gerando-se uma função auxiliar $\mathcal{F}'$ como definido pela Equação 3.

$$\mathcal{F}'(s \in S) = min_{x \in \mathcal{D}(s)}\{\mathcal{F}(x)\} \tag{3}$$

De forma análoga, um componente de diversidade $\mathcal{M}'(s)$ pode ser adaptado de ou-

tros propostos na literatura da forma $\mathcal{M}(x) \rightarrow \mathbb{R}$, utilizando o *melhor valor* encontrado entre todas soluções decodificadas para a atribuição. Porém, deve-se tomar o cuidado de não incluir no processo soluções completas geradas a partir uma mesma solução incompleta original, pois haveria uma penalização da solução sobre ela própria. Esta situação é ilustrada pela Figura 2, na qual uma solução "círculo" (gerada a partir de uma determinada solução incompleta) não é comparada a outros círculos, somente a soluções "quadrado" (geradas a partir de outras soluções incompletas).

**Figura 2. Cálculo de diversidade não deve incluir soluções completas geradas a partir de uma mesma solução incompleta, para evitar auto-penalizações. Quadrados e círculos simbolizam pontos no espaço $F$ de valores objetivos**




Desta maneira, o componente *crowding distance* do NSGA-II (linha 10 do Algoritmo 2) é aproveitado no algoritmo apresentado na próxima seção.

## 4 Matheurística inspirada no NSGA-II

A matheurística proposta é inspirada na metaheurística NSGA-II (Algoritmo 2) juntamente com o processo de decodificação exato de programação dinâmica (Algoritmo1). A técnica desenvolvida é apresentada no Algoritmo 3.

**Algoritmo 3**: Matheurística Proposta

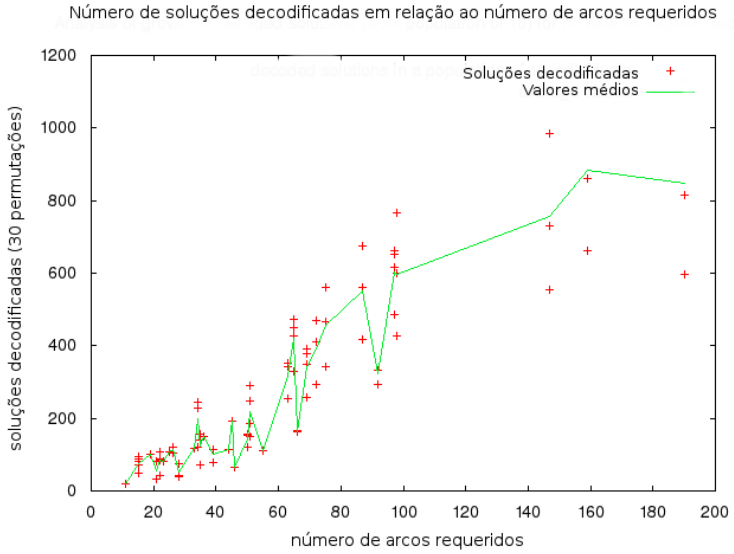
$P_S^0 \leftarrow$ população inicial aleatória
$N \leftarrow |P_S^0|$
$P_X^0 \leftarrow$ `Decodificação`$(P_S^0)$
$Q_S^0 \leftarrow$ `GerarFilhos`$(P_S^0)$
$Q_X^0 \leftarrow$ `Decodificação`$(Q_S^0)$
$t \leftarrow 1$
**enquanto** *critério de parada não satisfeito* **faça**
  $R_S^t \leftarrow P_S^{t-1} \cup Q_S^{t-1}$
  $R_X^t \leftarrow P_X^{t-1} \cup Q_X^{t-1}$
  $F \leftarrow$ `NonDominatedSort`$(R_X^t)$
  `AtribuiMelhorFitness`$(F,\ R_S^t,\ R_X^t)$
  $P_S^t \leftarrow \{\}$
  $P_X^t \leftarrow \{\}$
  $i \leftarrow 1$
  **while** $|P_S^t| + |F_i| \leq N$ **do**
    `CrowdingDistance`$(F_i,\ R_X^t)$
    `AtribuiMelhorDiversidade`$(F_i,\ R_S^t,\ R_X^t)$
    $P_S^t \leftarrow P_S^t \cup F_i[1 \ldots (N - |P_S^t|)]$
  **end**
  $Q_S^t \leftarrow$ `GerarFilhos`$(P_S^t)$
  $Q_X^t \leftarrow$ `Decodificação`$(Q_S^t)$
  $t \leftarrow t + 1$
**fim**
**return** $P_S^t$

O Algoritmo 3 segue a estrutura básica do NSGA-II incluindo a população $P_X$ de soluções decodificadas pelo procedimento `Decodificação`, sendo que esta população é atualizada sempre após operações na população de permutações $P_S$. Todas operações que alteram soluções são feitas sobre $P_S$, como a população inicial gerada aleatoriamente e a geração de novos indivíduos com base na população anterior (método `GerarFilhos`). Tal método utiliza uma taxa fixa $\alpha$ de mutação utilizando movimentos clássicos da literatura de roteamento de veículos: troca e realocação de elementos na permutação, também operação de 2-Opt (Lacomme *et al.*, 2004). Ainda no procedimento `GerarFilhos`, o operador `OX` foi utilizado para recombinação de pais, após um torneio binário clássico. Já os procedimentos nativos do NSGA-II chamados `NonDominatedSort` e `CrowdingDistance` operam sobre a população decodificada, sendo a população de permutações atualizada em seguida pelos métodos `AtribuiMelhorFitness` e `AtribuiMelhorDiversidade`, respectivamente, de acordo com as questões apresentadas na Seção 3.

## 5 Resultados Preliminares

Testes computacionais foram efetuados para analisar o crescimento do número de soluções decodificadas, em relação ao número de arestas requeridas. O conjunto de testes `gdb` proposto por Golden *et al.* (1983) e utilizado por Lacomme *et al.* (2006) consiste de 23 problemas-teste, com 7–27 nós e 11–55 arestas requeridas. Já o conjunto de testes `egl` de Belenguer e Benavent (2003), consiste de 77–140 nós e 98–190 arestas, sendo bem maior do que o conjunto `gdb`. A Figura 3 mostra o grande crescimento do número de soluções decodificadas de acordo com o tamanho do problema-teste, considerando uma população de 30 indivíduos.

**Figura 3. Crescimento do número de soluções decodificadas em relação ao número de arestas requeridas, considerando população de 30 indivíduos**

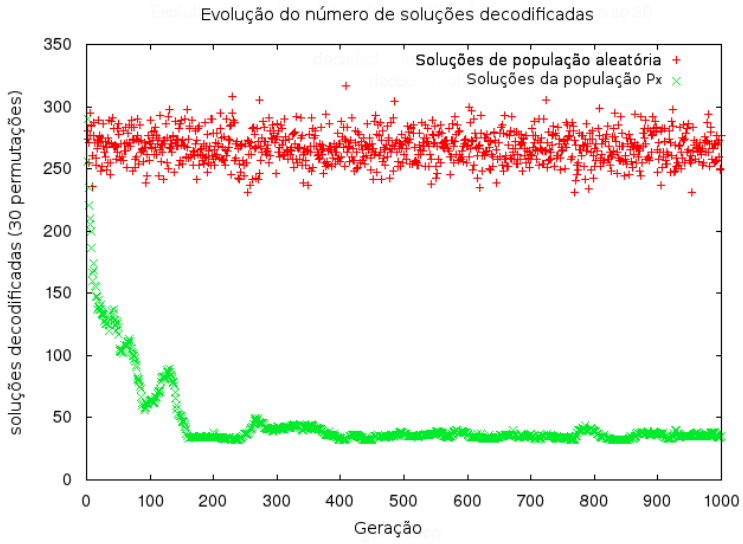



Na Figura 3, para problemas com 20 arestas requeridas, o número de soluções decodificadas é de aproximadamente 90 para toda a população, indicando uma média de três soluções decodificadas para cada permutação de entrada. Já para problemas maiores com 160 arestas não requeridas, o número de soluções decodificadas chega a 900, também

para 30 indivíduos, indicando uma média de 30 soluções decodificadas por permutação de entrada, ou seja, como a complexidade do algoritmo depende da população de soluções decodificadas e não das permutações de entrada, um pequeno aumento na população de permutações pode acarretar em uma grande sobrecarga computacional para problemas grandes.

Outro experimento efetuado buscou analisar se o número de soluções decodificadas é sensível à qualidade das permutações de entrada. A Figura 4 apresenta os resultados deste experimento para um problema-teste de tamanho grande com 120 arestas requeridas, limitando a execução da matheurística proposta a 1000 gerações.

**Figura 4. Permutações de melhor qualidade geram menos soluções decodificadas**




Na Figura 4 pode-se observar que enquanto o número de soluções decodificadas partindo de permutações aleatórias continua alto (em vermelho), o número de soluções decodificadas da população $P_X$ (em verde) cai até estagnar aproximadamente na geração 170, indicando que seus indivíduos estão sendo aprimorados pelos operadores de mutação e cruzamento. Porém, a partir da geração 170 o número de soluções decodificadas é estabilizado e indica que, embora de melhor qualidade, as permutações de entrada começam a tocar a fronteira de viabilidade do problema, havendo assim menos "espaço" para o decodificador exato encontrar opções variadas de rotas sujeitas à capacidade dos veículos.

Testes computacionais com os problemas-teste `gdb` foram efetuados em um Intel 2.3GHz Core i7 (somente um *core* foi utilizado), 8GB de RAM linux kernel 3.0, limitando o algoritmo a 1000 gerações sem melhora, com população de 80 permutações e $\alpha = 70\%$ para taxa de mutação, além de todas configurações do algoritmo apresentadas anteriormente. A Tabela 1 apresenta uma comparação da matheurística proposta com o NSGA-II de Lacomme *et al.* (2006). Na tabela, as colunas: $LI_1$ e $LI_2$ apresentam limites inferiores para os dois objetivos calculados por Belenguer e Benavent (2003) e Lacomme *et al.* (2006), respectivamente; $Melhor_1$ são os melhores valores da literatura para o primeiro objetivo; $F_1$ e $F_2$ são os melhores resultados do algoritmo para os dois objetivos; $Gap_1$ e $Gap_2$ apresentam a diferença percentual entre o melhor valor e o limite inferior; $M_1$ e $M_2$ apresentam a melhoria percentual entre a matheurística desenvolvida e o algoritmo de Lacomme *et al.* (2006). Valores marcados em negrito são valores ótimos em que o limite

inferior foi alcançado.

**Tabela 1. Comparação com resultados de Lacomme *et al.* (2006)**

| Problema-teste | | | | Lacomme *et al.* (2006) | | | | Matheurística proposta | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $LI_1$ | $LI_2$ | $Melhor_1$ | $F_1$ | $Gap_1(\%)$ | $F_2$ | $Gap_2(\%)$ | $F_1$ | $Gap_1(\%)$ | $F_2$ | $Gap_2(\%)$ | $M_1(\%)$ | $M_2(\%)$ |
| gdb01 | **316** | **63** | **316** | **316** | 0,0 | **63** | 0,0 | **316** | 0,0 | **63** | 0,0 | 0,0 | 0,0 |
| gdb02 | **339** | **59** | **339** | **339** | 0,0 | **59** | 0,0 | **339** | 0,0 | **59** | 0,0 | 0,0 | 0,0 |
| gdb03 | **275** | **59** | **275** | **275** | 0,0 | **59** | 0,0 | **275** | 0,0 | **59** | 0,0 | 0,0 | 0,0 |
| gdb04 | **287** | **64** | **287** | **287** | 0,0 | **64** | 0,0 | **287** | 0,0 | **64** | 0,0 | 0,0 | 0,0 |
| gdb05 | **377** | **64** | **377** | **377** | 0,0 | **64** | 0,0 | **377** | 0,0 | **64** | 0,0 | 0,0 | 0,0 |
| gdb06 | **298** | **64** | **298** | **298** | 0,0 | **64** | 0,0 | **298** | 0,0 | **64** | 0,0 | 0,0 | 0,0 |
| gdb07 | **325** | **57** | **325** | **325** | 0,0 | 61 | 7,0 | **325** | 0,0 | **57** | 0,0 | 0,0 | 7,0 |
| gdb08 | 344 | **38** | 348 | 350 | 1,7 | **38** | 0,0 | 350 | 1,7 | **38** | 0,0 | 0,0 | 0,0 |
| gdb09 | **303** | **37** | **303** | 309 | 2,0 | **37** | 0,0 | **303** | 0,0 | **37** | 0,0 | 1,9 | 0,0 |
| gdb10 | **275** | **39** | **275** | **275** | 0,0 | 54 | 38,5 | **275** | 0,0 | **39** | 0,0 | 0,0 | 38,5 |
| gdb11 | **395** | **43** | **395** | **395** | 0,0 | 64 | 48,8 | **395** | 0,0 | **43** | 0,0 | 0,0 | 48,8 |
| gdb12 | 448 | **93** | 458 | 458 | 2,2 | **93** | 0,0 | 458 | 2,2 | **93** | 0,0 | 0,0 | 0,0 |
| gdb13 | **536** | **128** | **536** | 544 | 1,5 | **128** | 0,0 | 544 | 1,5 | **128** | 0,0 | 0,0 | 0,0 |
| gdb14 | **100** | **15** | **100** | **100** | 0,0 | 17 | 13,3 | **100** | 0,0 | **15** | 0,0 | 0,0 | 13,3 |
| gdb15 | **58** | **8** | **58** | **58** | 0,0 | 13 | 62,5 | **58** | 0,0 | **8** | 0,0 | 0,0 | 62,5 |
| gdb16 | **127** | **14** | **127** | **127** | 0,0 | 19 | 35,7 | **127** | 0,0 | **14** | 0,0 | 0,0 | 35,7 |
| gdb17 | **91** | **9** | **91** | **91** | 0,0 | 15 | 66,7 | **91** | 0,0 | **9** | 0,0 | 0,0 | 66,7 |
| gdb18 | **164** | **19** | **164** | **164** | 0,0 | 27 | 42,1 | **164** | 0,0 | **19** | 0,0 | 0,0 | 42,1 |
| gdb19 | **55** | **17** | **55** | **55** | 0,0 | **17** | 0,0 | **55** | 0,0 | **17** | 0,0 | 0,0 | 42,1 |
| gdb20 | **121** | **20** | **121** | **121** | 0,0 | **20** | 0,0 | **121** | 0,0 | **20** | 0,0 | 0,0 | 0,0 |
| gdb21 | **156** | **15** | **156** | **156** | 0,0 | 22 | 46,7 | **156** | 0,0 | **15** | 0,0 | 0,0 | 46,7 |
| gdb22 | **200** | **12** | **200** | **200** | 0,0 | 20 | 66,7 | **200** | 0,0 | **12** | 0,0 | 0,0 | 66,7 |
| gdb23 | **233** | **13** | **233** | 235 | 0,9 | 20 | 53,8 | 235 | 0,9 | **13** | 0,0 | 0,0 | 53,8 |
| Média | | | | | 0,4 | | 20,9 | | 0,3 | | 0,0 | 0,1 | 20,9 |

Pela Tabela 1, pode-se observar que o algoritmo proposto encontra soluções melhores que Lacomme *et al.* (2006) para todas os problemas-teste, tendo o resultado mais expressivo para o segundo objetivo com uma melhoria média de 20,9%. Para o primeiro objetivo, a melhoria média foi de apenas 0,1%, embora não haja mais muito espaço para melhora nesse conjunto de problemas visto que apenas `gdb08` e `gdb12` seguem em aberto. O resultado expressivo no segundo objetivo também já era esperado visto que o decodificador exato proposto é capaz de encontrar elementos que representem toda a frente de pareto, indicando que este objetivo foi um pouco negligenciado no trabalho de Lacomme *et al.* (2006).

A Tabela 2 apresenta o tempo computacional do algoritmo proposto e o compara também a outros trabalhos anteriores. Na tabela, a coluna "Algoritmo" apresenta o autor do trabalho; "Pareto" indica o número médio de elementos na frente de pareto apresentada (somente para versão biobjetiva); *Gap* $LI_1$ e *Gap* $LI_2$ são as diferenças percentuais entre as soluções encontradas e os respectivos limites inferiores; *Ótimos* $LI_1$ e *Ótimos* $LI_2$ apresentam o número de soluções ótimas encontradas para cada objetivo; "Tempo" apresenta o tempo computacional do algoritmo, em segundos.

**Tabela 2. Comparação de tempo computacional com demais algoritmos da literatura**

| Algoritmo | Pareto | *Gap* $LI_1$ | *Ótimos* $LI_1$ | *Gap* $LB_2$ | *Ótimos* $LB_2$ | Tempo(s) |
|---|---|---|---|---|---|---|
| Hertz *et al.* (2000) | | 0,5 | 18 | | | 1,6* |
| Lacomme *et al.* (2004) | | 0,2 | 21 | | | 0,9* |
| Lacomme *et al.* (2006) | 3,4 | 0,4 | 18 | 20,9 | 12 | 4,3* |
| Matheurística | 6,4 | 0,3 | 19 | 0,0 | 23 | 17,1 |

* tempo foi reduzido devido às máquinas diferentes, levando em consideração um fator de 216/631=0.34

Pela Tabela 2, percebe-se que o algoritmo proposto encontra em média 6,4 soluções na frente de pareto, enquanto o algoritmo biobjetivo de Lacomme *et al.* (2006) encontra

apenas 3,4. Porém, essa maior variedade de soluções faz com que a matheurística trabalhe com um número demasiado de soluções decodificadas, aumentando seu tempo computacional. Quando se considera apenas o primeiro objetivo, o trabalho de Lacomme *et al.* (2004) ainda é referência, visto se baixo tempo computacional e capacidade de encontrar 21 soluções ótimas (de um conjunto com 23 problemas). Outros testes foram efetuados com a matheurística proposta em problemas-teste maiores como o `egl`, porém o tempo computacional tem se mostrado proibitivo nestes casos e estratégias de redução do número de soluções decodificadas estão sendo desenvolvidas para atingir esta meta.

## 6 Conclusões

Neste trabalho, foi apresentada uma matheurística para o CARP biobjetivo, problema com diversas aplicações práticas que consiste em atender um conjunto de arestas com um conjunto de veículos de capacidade limitada. Os objetivos considerados são de minimizar o somatório das distâncias percorridas em todas as rotas e também minimizar o tamanho da maior rota. O CARP biobjetivo é $\mathcal{NP}$-Difícil e algumas abordagens heurísticas já foram apresentadas anteriormente na literatura. A matheurística desenvolvida consiste de uma metaheurística clássica multiobjetiva chamada NSGA-II, juntamente com uma função de decodificação exata em programação dinâmica. Visto que a função de decodificação é capaz de gerar um conjunto de solução para cada permutação de arestas dada como entrada, o arcabouço clássico multiobjetivo deve ser adaptado para que estratégias conhecidas de atribuição de *fitness* e manutenção de diversidade sejam reutilizadas. A matheurística proposta foi comparada a outros trabalhos da literatura, demonstrando uma boa capacidade de encontrar frentes de pareto diversas e de boa qualidade para ambos objetivos considerados. O tempo computacional do algoritmo ainda é um problema visto o grande número de soluções decodificadas, assim propõe-se como trabalho futuro desenvolver técnicas de filtragem e compressão do número de soluções decodificadas para viabilizar a aplicação do algoritmo em problemas de maior porte.

## 7 Agradecimentos

Os autores agradecem o apoio financeiro do CNPq e FAPERJ, bem como a bolsa de doutorado sanduíche da CAPES (processo 10381/11-3).

## Referências

Belenguer, José M. e Benavent, Enrique. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, v. 30, n. 5, p. 705 – 728. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/S0305-0548(02)00046-1. URL `http://www.sciencedirect.com/science/article/pii/S0305054802000461`.

Charnes, A. e Cooper, W. W. (1977). Goal programming and multiple objective optimization. *European Journal of Operational Research*, v. 1, n. 1, p. 39–45.

Deb, K.; Pratap, A.; Agarwal, S. e Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, v. 6, n. 2, p. 182–197. ISSN 1089-778X. doi: 10.1109/4235.996017.

Deb, Kalyanmoy; Sundar, J.; N, Udaya Bhaskara Rao e Chaudhuri, Shamik. (2006). Reference point based multi-objective optimization using evolutionary algorithms. *International Journal of Computational Intelligence Research*, p. 635–642. Springer-Verlag, (2006).

Dror, Moshe; Leung, JannyM.Y. e Mullaseril, PaulA. (2000). Livestock feed distribution and arc traversal problems. Dror, Moshe, editor, *Arc Routing*, p. 443–464. Springer US. ISBN 978-1-4613-7026-0. doi: 10.1007/978-1-4615-4495-1_12. URL http://dx.doi.org/10.1007/978-1-4615-4495-1_12.

French, S. (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-shop*. Ellis Horwood series in mathematics and its applications. E. Horwood. ISBN 9780853123644. URL http://books.google.fr/books?id=Ey9gRQAACAAJ.

Golden, B L e Wong, R T. (1981). Capacitated arc routing problems. *Networks*, v. 11, p. 305–315.

Golden, B.L.; DeArmon, JS e Baker, EK. (1983). Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research*, v. 10, n. 1, p. 47–59. URL http://www.sciencedirect.com/science/article/pii/0305054883900266.

Haimes, Y. Y.; Lasdon, L. S. e Wismer., D. A. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *Systems, Man and Cybernetics, IEEE Transactions on*, v. SMC-1, n. 3, p. 296–297. ISSN 0018-9472. doi: 10.1109/TSMC.1971.4308298.

Hertz, Alain; Laporte, Gilbert e Mittaz, Michel. (2000). A Tabu Search Heuristic for the Capacitated Arc Routing Problem. *Operations Research*, v. 48, n. 1, p. 129–135. URL http://www.jstor.org/stable/222920.

Lacomme, P.; Prins, C. e Sevaux, M. December(2006). A genetic algorithm for a bi-objective capacitated arc routing problem. *Computers & Operations Research*, v. 33, n. 12, p. 3473–3493. ISSN 03050548. doi: 10.1016/j.cor.2005.02.017. URL http://linkinghub.elsevier.com/retrieve/pii/S0305054805000730.

Lacomme, Philippe; Prins, Christian e Ramdane-Cherif, Wahiba. October(2004). Competitive Memetic Algorithms for Arc Routing Problems. *Annals of Operations Research*, v. 131, n. 1-4, p. 159–185. ISSN 0254-5330. doi: 10.1023/B:ANOR.0000039517.35989.6d. URL http://link.springer.com/10.1023/B:ANOR.0000039517.35989.6d.

Lannez, S.; Artigues, C.; Damay, J. e Gendreau, M. (2010). Column generation heuristic for a rich arc routing problem. Erlebach, Thomas e Lübbecke, Marco, editors, *10th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS)*, volume 14 of *OpenAccess Series in Informatics (OASIcs)*, p. 130–141, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL http://dx.doi.org/10.4230/OASIcs.ATMOS.2010.130. ISBN 978-3-939897-20-0.

Stern, Helman I e Dror, Moshe. (1979). Routing electric meter readers. *Computers & Operations Research*, v. 6, n. 4, p. 209–223.

Talbi, El-Ghazali. (2009). *Metaheuristics: From Design to Implementation*. Wiley Publishing. ISBN 0470278587, 9780470278581.

Zitzler, E. e Thiele, L. November(1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *Trans. Evol. Comp*, v. 3, n. 4, p. 257–271. ISSN 1089-778X. doi: 10.1109/4235.797969. URL http://dx.doi.org/10.1109/4235.797969.

Zitzler, Eckart e Künzli, Simon. (2004). Indicator-based selection in multiobjective search. *in Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII*, p. 832–842. Springer, (2004).