Uma Abordagem Populacional para o Algoritmo de Busca em Vizinhança Variável Aplicado em Otimização Contínua

Wesklei Migliorini[1] **, Rafael Stubs Parpinelli**[1]

[1]Departamento de Ciência da Computação - Universidade do Estado de Santa Catarina (UDESC)
Caixa Postal 89219-710 – Joinville– SC – Brasil

wesklei.m@gmail.com, parpinelli@joinville.udesc.br

Abstract. *This paper presents a population-based approach to the Variable Neighbourhood Search algorithm, named PRVNS. The main contribution of the proposed algorithm, in addition to evolve a population of individuals, is that each individual adapts its neighborhood variations autonomously. This autonomous neighborhood control allows individuals to intensify or diversify the search for promising regions in the solution space during the optimization process. Hence, each individual adapts its behavior according to the region in which it is located. This work focuses on optimization problems with continuous domain. Several benchmark functions with high dimensionality ($D = 250$) were used. Results were obtained and compared with the non-populational VNS approach and with the Differential Evolution algorithm. Results suggest that the proposed approach is a promising and competitive alternative to continuous optimization.*

Resumo. *Este trabalho apresenta uma abordagem populacional para o algoritmo de Busca em Vizinhança Variável, denominado PRVNS. A principal contribuição do algoritmo proposto, além de evoluir uma população de indivíduos, é que cada indivíduo adapta suas variações de vizinhança de maneira autônoma. Este controle autônomo de vizinhança permite aos indivíduos intensificar ou diversificar a busca por regiões promissoras no espaço de soluções durante o processo de otimização. Cada indivíduo adapta seu comportamento de acordo com a região em que se encontra no espaço de soluções. Este trabalho tem como foco de aplicação problemas de otimização com domínio contínuo. Foram utilizadas várias funções benchmark com alta dimensionalidade ($D = 250$). Resultados foram obtidos e comparados com a abordagem não-populacional do VNS e com o algoritmo populacional de Evolução Diferencial. Resultados sugerem que a abordagem proposta é uma alternativa promissora e competitiva para otimização contínua.*

1. Introdução

O algoritmo de Busca em Vizinhança Variável (*Variable Neighbourhood Search - VNS*) é uma meta-heurística de melhoras iterativas que vem sendo aplicada com sucesso na resolução de problemas tanto em domínios discretos, como problemas de atribuição de tarefas [Kratica et al. 2010] e sequenciamento de produção [Ribeiro et al. 2008], quanto

em domínios contínuos, como no treinamento de redes neurais [Alba and Martí 2006] e máquina de vetores de suporte [Carrizosa et al. 2012]. Sua principal característica é modificar iterativamente uma única solução corrente, utilizando variações de vizinhança no decorrer da busca.

Existem algumas variações do *VNS* mas sua versão mais canônica é o *VNS* Reduzido (*RVNS*) que utiliza uma solução candidata durante a busca variando a sua vizinhança na tentativa de encontrar melhores soluções. Essencialmente, o algoritmo *VNS* não utiliza uma abordagem populacional. Um algoritmo populacional possui várias soluções candidatas (indivíduos) que são otimizadas em paralelo. Cada indivíduo é iniciado de modo aleatório em diferentes regiões do espaço de busca, permitindo explorar diferentes regiões, aumentando a diversidade da busca. Alguns exemplos de algoritmos populacionais são a Otimização por Enxame de Partículas (*PSO*) [Kennedy 2010], Evolução Diferencial (*DE*) [Price et al. 2006] e Otimização por Colônia de Formigas (*ACO*) [Dorigo and Birattari 2010].

Algumas propostas populacionais para o *VNS* foram encontradas na literatura, todas abordando problemas discretos. Os principais pontos a serem considerados nessas propostas são as rotinas de perturbação e a troca de soluções. O trabalho de [Ng 2010] apresenta uma proposta para o problema *Minimum Shift Design* e utiliza uma abordagem de perturbação sem informação da população, gerando aleatoriamente um vizinho da solução sendo avaliada, usando a amplitude de vizinhança atual. Esta amplitude é a mesma usada para todos os indivíduos da população. A atualização das soluções usa uma estratégia gulosa trocando sempre a pior solução de toda a população. Já em [Hsiao et al. 2012] são abordados quatro problemas: *Max-SAT*, *Bin Packing*, *Flow Shop Scheduling* e *Personnel Scheduling* e o trabalho apresenta uma proposta de *hiper*-heurística que utiliza um *VNS* populacional. Durante a etapa de perturbação é realizada uma mutação entre indivíduos. A troca de solução só é feita pela melhor solução, e caso não seja melhor é feito um torneio de dois indivíduos aleatórios da população substituindo o pior. No trabalho de [Wang and Tang 2009] o problema abordado é *Single Machine Total Weighted Tardiness Problem* e a perturbação é feita selecionando m candidatos aleatórios que são combinados através de uma estratégia gulosa que seleciona sempre a melhor combinação das soluções encontradas e então é gerada uma nova solução que considera a amplitude de vizinhança atual. Esta perturbação utiliza a mesma amplitude de vizinhança para toda a população. Para atualizar as soluções, se a solução gerada pela perturbação for a melhor já encontrada, o pior da população é substituído. Caso contrário é substituída a solução mais distante da melhor solução, dada uma regra de distância estabelecida.

Este trabalho propõe uma versão populacional (*PRVNS*) do algoritmo *RVNS* para problemas de otimização contínua utilizando a amplitude de vizinhança no processo de perturbação [Gendreau and Potvin 2010]. Sendo um algoritmo populacional, pode-se usar várias soluções candidatas para explorar o espaço de soluções. Nesta proposta cada indivíduo define as variações de vizinhança de forma independente. O controle de amplitude separada para cada indivíduo permite intensificar ou diversificar a busca por regiões promissoras no espaço de soluções de maneira heterogênea. De maneira geral, uma maior amplitude caracteriza a diversificação, enquanto que uma amplitude menor reforça a intensificação da busca [Sheikh Rajab 2012]. Desta maneira, cada indivíduo

na população é responsável por coordenar seu comportamento durante o processo de otimização. Esta é a principal característica que difere a abordagem aqui proposta das demais abordagens encontradas. A troca de soluções é feita através da estratégia gulosa, só atualizando a solução corrente se a nova solução for melhor. Quando ocorre a troca de solução, a amplitude é diminuída para intensificar a exploração na vizinhança e quando não ocorre melhora a amplitude aumenta tentando explorar soluções mais distantes.

A estrutura deste trabalho segue na Seção 2 com uma revisão bibliográfica sobre os conceitos que envolvem o algoritmo *VNS*; a Seção 3 mostra o desenvolvimento do algoritmo proposto *PRVNS*; a Seção 4 discute os experimentos realizados; a análise dos resultados é feita na Seção 5; por fim as conclusões do trabalho são apresentadas na Seção 6, juntamente com os trabalhos futuros.

2. Busca em Vizinhança Variável

O algoritmo de Busca em Vizinhança Variavel (*Variable Neighbourhood Search - VNS*) é um algoritmo meta-heurístico proposto por Mladenović and Hansen em 1997 que modifica uma solução corrente explorando vizinhos na tentativa de encontrar melhores soluções. No *VNS*, a amplitude da busca varia dinamicamente de acordo com a dificuldade de melhora da solução corrente [Gendreau and Potvin 2010].

Uma estrutura de vizinhança pode ser denotada por $\mathcal{N}_k(\vec{x})$, sendo $\vec{x}$ a solução corrente e k o índice da estrutura de vizinhança sendo explorada (k varia entre k_1 e k_{max}). O vetor $\vec{x}$ é um vetor solução *d*-dimensional ($\vec{x} = [x_1, x_2, ..., x_d]$) e entende-se por vizinhos de $\vec{x}$ os vetores próximos a ele obedecendo a métrica e amplitude de vizinhança utilizada. Para delimitar os vizinhos em uma região pode-se usar uma amplitude ou raio definido por $r_k(k = 1, .., k_{max})$ em que ao longo do processo de busca, k pode variar, ampliando ou contraindo a amplitude de vizinhos de $\vec{x}$ [Mladenović et al. 2008].

Para a estrutura $\mathcal{N}_k(\vec{x})$ é possível utilizar uma ou mais métricas sendo definidas por

$$\mathcal{N}_k(\vec{x}) = \{\vec{y} \in \mathcal{X} | r_{k-1} < \rho_k(\vec{x}, \vec{y}) \leq r_k\} \tag{1}$$

ou ainda

$$\mathcal{N}_k(\vec{x}) = \{\vec{y} \in \mathcal{X} | \rho_k(\vec{x}, \vec{y}) \leq r_k\} \tag{2}$$

Estas métricas são as representadas na Figura 1 e definem a geometria da estrutura $\mathcal{N}_k(\vec{x})$ no espaço de busca onde $\rho_k(\vec{x}, \vec{y})$ é a distância entre duas soluções e é dada por

$$\rho_k(\vec{x}, \vec{y}) = (\sum_{i=1}^{d} |x_i - y_i|^p)^{\frac{1}{p}} \qquad (1 \leq p \leq \infty)$$

ou

$$\rho_k(\vec{x}, \vec{y}) = \max_{0 \leq i \leq n} |x_i - y_i|, p = \infty$$

onde p define a métrica ℓ_p: Manhattan (ℓ_1), Euclidiana (ℓ_2) ou Chebyshev (ℓ_∞). A utilização destas métricas pode se dar de forma heterogênea usando mais de uma métrica e variando-as conforme a execução, ou de forma homogênea mantendo a mesma métrica ao longo de toda execução. Sua escolha e como serão usadas depende da descrição do problema [Carrizosa et al. 2012].

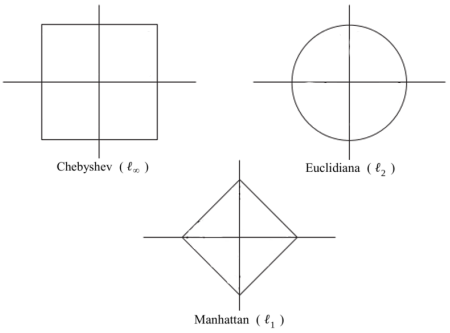


Figura 1. Tipos de métricas

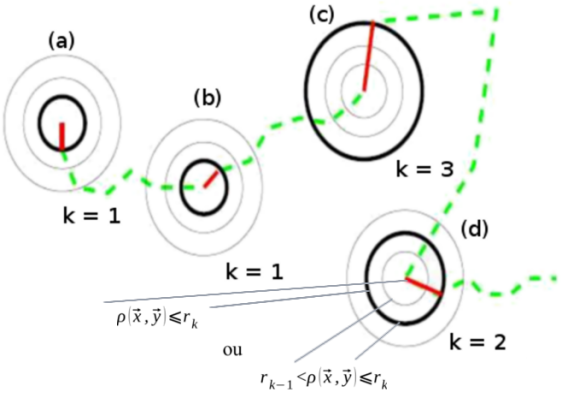


Figura 2. Busca em vizinhança variável com ℓ_2. Adaptado de [Sheikh Rajab 2012]

As estruturas de vizinhança para a mesma solução respeitam a relação $\mathcal{N}_1(\vec{x}) \subseteq \mathcal{N}_2(\vec{x}) \subseteq ... \subseteq \mathcal{N}_{k_{max}}(\vec{x})$ pois $\mathcal{N}_k(\vec{x})$ aumenta sua amplitude expandindo a estrutura atual para uma nova vizinhança. Esse comportamento é ilustrado na Figura 2 usando a métrica Euclidiana (ℓ_2). No passo **(a)**, k inicia em 1 e $\mathcal{N}_k(\vec{x})$ pode ser visto pelo círculo em negrito, quando o *VNS* encontra uma solução melhor ele a assume como solução corrente mantendo ou reiniciando o valor de $k = 1$ como visto no passo **(b)**. Em **(c)**, k foi ampliado até chegar no k_{max} que neste caso é 3 pois não encontrou melhor solução com valores menores de k. **(d)** demonstra a relação $\rho_k(\vec{x}, \vec{y})$ que pode ter dois comportamentos como na Equação 1 e 2 [Sheikh Rajab 2012].

O *VNS* possui algumas variações encontradas na literatura e sua forma mais canônica é o *VNS* Reduzido (*RVNS*). Seu algoritmo consiste apenas no processo de perturbação e troca de vizinhança até alcançar a condição de parada, como observado na Figura 2 e no Algoritmo 1 em que uma única solução é otimizada iterativamente aplicando-se as vizinhanças variáveis. Para executar o *RVNS* primeiramente deve-se definir a métrica, os valores da estrutura de vizinhança $\mathcal{N}_k$ de k = 1 à k_{max} e gerar um vetor inicial $\vec{x}$ pertencente ao espaço de busca da função. A condição de parada do *RVNS* utiliza um contator que tem seu limite definido previamente, e pode ser o número de avaliações da função objetivo ou o tempo de *CPU* [Alba and Martí 2006]. O Algoritmo 1 detalha a abordagem *RVNS* [Gendreau and Potvin 2010].

A próxima seção descreve a abordagem populacional proposta neste trabalho, o *PRVNS*.

Algoritmo 1: Algoritmo RVNS

```
1  input : N_k, LB, UB, t_max, k_max
   output: Melhor solução encontrada x⃗
2  Definir a métrica e estrutura de vizinhança N_k,k = 1, ..., k_max;
3  Gerar aleatoriamente um ponto inicial x⃗;
4  Avaliar f(x⃗);
5  while t > t_max do
6      k ← 1 ;
7      while k ≤ k_max do
           // Perturba o ponto local gerando um
           novo candidato;
8          y⃗ ← x ;
9          i ← rand(1, d) // índice aleatório ;
10         y[i] ← rand(LB, UB) ;
11         Avaliar f(y⃗);
           // Troca a solução e a vizinhança ;
12         if f(y⃗) < f(x⃗) // Minimização ;
13         then
14             x⃗ ← y⃗ // Faz um movimento
15             k ← 1 // Volta a primeira amplitude de
                   vizinhança
16         else k++ // Amplia a vizinhança ;
17     t++ // Incrementa o contador de iteração
18 return x⃗;
```

3. Abordagem Populacional

O Algoritmo de Busca em Vizinhança Variável Reduzido baseado em População (*Population-based Reduced Variable Neighbourhood Search - PRVNS*) faz uso não somente de uma única solução candidata para vasculhar o espaço de soluções, mas sim de um conjunto de possíveis soluções, chamado de população. O algoritmo se baseia no comportamento do *VNS* canônico (*RVNS*) e por isso é denotado *PRVNS*. No *RVNS* tem-se a estrutura denotada por $\mathcal{N}_k(\vec{x})$ sendo que $\vec{x}$ é um indivíduo e k sempre referencia este mesmo indivíduo. Já na versão populacional tem-se um valor de k associado à cada indivíduo, podendo assim assumir valores distintos, o que pode ser interessante para a diversificação da população. Desta maneira, pode-se definir a estrutura de vizinhança populacional por $\mathcal{N}_{k_i}(\vec{x_i})$ em que k_i é o índice da estrutura de vizinhança atual para o indivíduo $\vec{x_i}$ da população. Este índice varia entre k_{i_1} e $k_{i_{max}}$. Modificando o respectivo k_i de um indivíduo é possível expandir ou contrair a amplitude de vizinhança. O Algoritmo 2 detalha a abordagem *PRVNS* usando a definição de estrutura de vizinhança populacional.

O Algoritmo 2 define o tamanho da população com o parâmetro n, a probabilidade de ocorrer perturbação através do parâmetro PC, a condição de parada com o valor máximo de gerações G_{max}, o número máximo de vizinhanças k_{max} e o valor de amplitude atribuído a cada vizinhança r_{k_i}. No laço da linha 2, o algoritmo inicializa a população com soluções candidatas $\vec{x_i}$, definindo sua vizinhança inicial como $k_i = 1$ e avaliando sua função objetivo. A seguir o algoritmo executa a iteração da linha 6 até a condição de parada ser alcançada. Neste laço estão as principais etapas do Algoritmo 2: perturbação e troca da amplitude de vizinhança. A perturbação ocorre entre as linhas 8 e 14. No *RVNS* a perturbação é feita modificando um índice de $\vec{x}$ para gerar $\vec{y}$. No *PRVNS* é utilizado a informação da solução de outros indivíduos da população para compor o vetor perturbado (linha 8). O controle de expansão e contração faz analogia com a amplitude de vizinhança do *VNS* e é visto na linha 8 onde dois indivíduos aleatórios são selecionados e usados na linha 12 junto ao valor de $rand(-r_{k_i}, r_{k_i})$ que controla a amplitude de vizinhança. O valor de p selecionado na linha 9 garante que pelo menos um índice de $x_{i,j}$ seja perturbado,

Algoritmo 2: Esquema geral do algoritmo PRVNS

```
1
  input : n, PC, G_max, k_max, r_{k_i} (1 ≤ i ≤ k_max)
  output: Melhor solução encontrada s⃗
2 for (i=0; i < n; i++) do
3     Inicializar x⃗_i aleatoriamente
4     Inicializar vizinhança k_i = 1
5     Avaliar função objetivo f(x⃗_i)
6 while (iter < G_max) do
      // Etapa populacional
7     for (i=0; i < n; i++) do
          // Etapa de perturbação
8         Selecionar índices s_1, s_2 ∈ n aleatoriamente, s_1 / s_2 / i
9         Selecionar dimensão p ∈ d                       =     =
10        for (j=0; j < d; j++) do
11            if (j == p) ∨ (rand(0,1) ≤ PC) then
12                y_j = x_{s_2,j} + rand(−r_{k_i}, r_{k_i}) ∗ (x_{s_1,j})
13            else
14                y_j = x_{i,j}
          // Etapa de avaliação e troca de
             amplitude da vizinhança
15        Avaliar f(y⃗)
16        if f(y⃗) < f(x⃗_i)// Minimização
17        then
18            x⃗_i ← y⃗ // Faz um movimento
19            k_i ← 1 // Volta a primeira amplitude de
                 vizinhança
20        else
              // Troca de amplitude
21            if (k_i < k_max) then
22                k_i + +
23    iter + +
24 s⃗ ← Encontrar melhor solução da população ;
25 return s⃗ ;
```

porém é mantido o parâmetro PC para controlar a probabilidade de perturbação. Entre as linhas 15 e 22 é feita a troca de vizinhança. Na linha 15 é realizada a avaliação da função objetivo e caso o valor de $\vec{y}$ seja melhor que a solução corrente, ele troca a solução e reinicia a amplitude de vizinhança k_i como visto na linha 18 e 19. Caso a solução gerada ($\vec{y}$) não seja melhor do que a solução corrente ($\vec{x_i}$), a amplitude de vizinhança é aumentada na linha 22 com o objetivo de tentar explorar regiões mais distantes. A amplitude é incrementada até que o valor de ki_{max} seja alcançado. O algoritmo encerra com a condição de parada G_{max} que é o número máximo de gerações.

4. Experimentos

Experimentos foram realizados utilizando os algoritmos *RVNS*, *PRVNS* e a Evolução Diferencial (*DE*). O *DE* é um algoritmo populacional que utiliza um peso fixo F para perturbação fazendo a diferença entre três indivíduos selecionados aleatoriamente. Por também usar um valor de peso na influencia da perturbação, optou-se em utilizar o *DE* nos experimentos como algoritmo de comparação ao *PRVNS*. Os algoritmos foram aplicados a um conjunto de 10 funções de *benchmark* para otimização contínua, descritas na Tabela 1. A tabela mostra a sigla de cada função usada no trabalho, bem como o nome, a equação matemática que a representa, o domínio e o ótimo global para minimização da função. Todos os experimentos foram executados utilizando computadores *AMD Phenom II X4 B93* com *4GB* de memória e ambiente *Linux 64 bits*. Para todos os algoritmos foram utilizados 500.000 avaliações da função objetivo em cada execução, *DE* e *PRVNS* utilizaram uma população de 50 indivíduos cada um. Foram realizadas 30 execuções de cada função com dimensão *d = 250* para cada abordagem. Optou-se por uma alta dimen-

sionalidade nas funções para poder avaliar o desempenho dos algoritmos em problemas com alto nível de complexidade. O *RVNS* e *PRVNS* utilizam a métrica ℓ_∞ e a relação da Equação 2. O *RVNS* usa $k_{max} = 5$ com valores: $0.1, 0.28, 0.78, 2, 19$ e 6.14 obtidos de uma progressão geométrica de razão 2.8 definida de forma empírica. O *PRVNS* utiliza valores distribuídos uniformemente entre 0 e 1: $0.1, 0.3, 0.5, 0.7$ e 0.9 também com $k_{max} = 5$ de forma empírica. Os demais parâmetros do *DE* seguem os valores encontrados na literatura para $PC = 0.9$ e $F = 0.4717$ [Pedersen 2010]. O $PC = 0.9$ também foi utilizado para o *PRVNS*.

Tabela 1. Funções avaliadas no experimento

Função	Nome	Definição	Domínio	Ótimo Global
$f_1(\vec{x})$	Rastrigin	$\sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$-5.12 \leq x_i \leq 5.12$	$f_1(\vec{0}) = 0$
$f_2(\vec{x})$	Schaffer F7	$[\frac{1}{n-1}\sqrt{s_i} \cdot (sin(50.0s_i^{\frac{1}{5}}) + 1)]^2 s_i = \sqrt{x_i^2 + x_{i+1}^2}$	$-100 \leq x_i \leq 100$	$f_2(\vec{0}) = 0$
$f_3(\vec{x})$	Ackley	$-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	$-32 \leq x_i \leq 32$	$f_3(\vec{0}) = 0$
$f_4(\vec{x})$	Rosenbrock	$\sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$-30 \leq x_i \leq 30$	$f_4(\vec{1}) = 0$
$f_5(\vec{x})$	Sphere	$\sum_{i=1}^{n} x_i^2$	$-5.12 \leq x_i \leq 5.12$	$f_5(\vec{0}) = 0$
$f_6(\vec{x})$	Schaffer F6	$\sum_{i=1}^{n-1}\left(0.5 + \frac{\sin^2(\sqrt{x_{i+1}^2 + x_i^2}) - 0.5}{(0.001(x_{i+1}^2 + x_i^2) + 1)^2}\right)$	$-100 \leq x_i \leq 100$	$f_6(\vec{0}) = 0$
$f_7(\vec{x})$	Levy	$\sin^2(\pi w_1)\sum_{i=0}^{d-1}(w_i - 1)^2[1 + 10sin^2(\pi w_i + 1)] + (w_d - 1)^2[1 + \sin^2(2\pi w_d)]$	$-10 \leq x_i \leq 10$	$f_7(\vec{1}) = 0$
$f_8(\vec{x})$	Zakharov	$\sum_{i=1}^{d} x_i^2 + (\sum_{i=1}^{d} 0.5ix_i)^2 + (\sum_{i=1}^{d} 0.5ix_i)^4$	$-5 \leq x_i \leq 10$	$f_8(\vec{0}) = 0$
$f_9(\vec{x})$	Schwefel 2.22	$\sum_{i=0}^{n} \lvert x_i \rvert + \prod_{i=0}^{n} \lvert x_i \rvert$	$-10 \leq x_i \leq 10$	$f_9(\vec{0}) = 0$
$f_{10}(\vec{x})$	Griewank	$\frac{1}{4000}(\sum_{i=1}^{n} x_i^2) - \left(\prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$	$-600 \leq x_i \leq 600$	$f_{10}(\vec{0}) = 0$

5. Resultados e Análises

Os resultados do experimento para as dez funções *bechmark* são mostrados na Tabela 2 com o número e nome de cada função, o valor de mínimo global e a média com desvio padrão sendo separadas para os algoritmos *RVNS*, *PRVNS* e *DE*, respectivamente. Em negrito esta ressaltado o algoritmo que obteve o melhor resultado para determinada função. Na parte inferior da tabela a quantidade de melhores soluções e de soluções ótimas encontradas foram sumarizadas para cada algoritmo. Na Tabela 2 o *PRVNS* foi o único algoritmo a encontrar a solução ótima em F3, F5 e F9 além de obter 5 melhores soluções em F1, F3, F5, F7 e F9, seguido do *DE* com 3 em F6, F8 e F10. O *RVNS* não obteve a melhor solução em nenhum caso. Na função F2 o *PRVNS* e o *DE* ficaram equivalentes quando analisado o desvio padrão mas ambos tiveram melhor solução que o *RVNS*, e em F4 o desvio padrão fez com que os três algoritmos ficassem equivalentes, mas o *DE* teve o menor desvio padrão.

O gráfico de convergência dos algoritmos *RVNS*, *PRVNS* e *DE* é mostrado na Figura 3 para cada uma das dez funções *benchmark*. O eixo y representa a geração e o eixo x representa o valor da média da melhor solução encontrada a cada execução do algoritmo na respectiva geração y. Para melhor visualização da convergência, em algumas funções foi utilizado a escala logarítmica denotada por LOG(Gerações) em y. No comportamento de convergência de um algoritmo, o desejável é manter o equilíbrio entre diversificação e intensificação da busca, objetivando manter uma boa diverssidade de soluções durante o processo de otimização, o que pode evitar ficar preso em pontos locais no espaço de soluções. A Figura 3 mostra que, em quase todas as funções, o *RVNS* não conseguiu convergir sua solução ao longo das gerações, com exceção da função F2 e F6 com uma pequena melhora no início. Esta verificação indica pouca habilidade do algoritmo sair de pontos locais no espaço de solução do problema. O *PRVNS* iniciou convergindo nas funções F8 e F10 porém nas últimas gerações teve pouca melhora. Nas demais funções

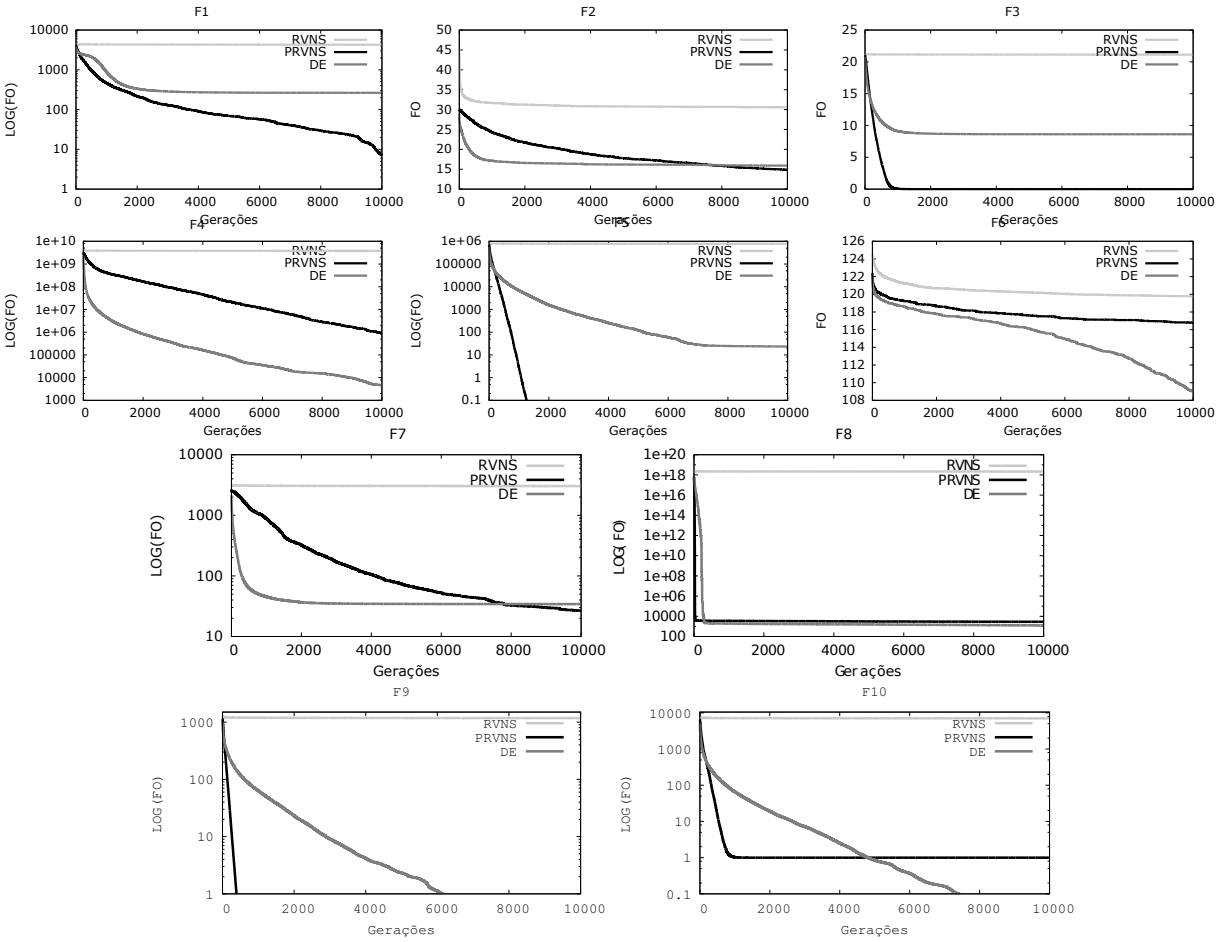
Tabela 2. Resultados obtidos para os experimentos com as funções (*d = 250*)

Número	Função	Mínimo		RVNS	PRVNS	DE
F1	*Rastrigin*	0	Média	4.601e+03	**0.05**	264.60
			Desv. P.	1.53e+02	**0.29**	32.64
F2	*Schaffer F7*	0	Média	45.82	14.81	15.91
			Desv. P.	9.65	1.45	0.59
F3	*Ackley*	0	Média	21.21	**0.00**	8.62
			Desv. P.	0.06	**0.00**	0.63
F4	*Rosenbrock*	0	Média	4.10e+09	9.23e+05	4.65e+03
			Desv. P.	3.47e+08	3.13e+06	6.62e+03
F5	*Sphere*	0	Média	8.38e+05	**0.00**	23.36
			Desv. P.	5.17e+04	**0.00**	118.60
F6	*Schaffer F6*	0	Média	122.52	116.76	**109.00**
			Desv. P.	0.90	0.88	**4.88**
F7	*Levy*	0	Média	3.29e+03	**21.39**	34.27
			Desv. P.	2.70e+02	**1.39**	4.91
F8	*Zakharov*	0	Média	2.16e+18	2582.43	**1234.27**
			Desv. P.	9.51e+17	177.27	**141.28**
F9	*Schwefel 2.22*	0	Média	1180.00	**0.00**	0.07
			Desv. P.	35.60	**0.00**	0.17
F10	*Griewank*	0	Média	7.05e+03	1.00	**0.06**
			Desv. P.	3.26e+02	0.00	**0.28**
Melhores soluções:				0	**5**	3
Ótimo global:				0	**3**	0

ele conseguiu manter uma boa convergência como visto em F0, F1, F2, F3, F4, F5, F6, F7 e F9. Em F3 e F5 o *PRVNS* teve uma rápida convergência, encontrando o ponto ótimo com poucas gerações.

Comparando o *PRVNS* com o *RVNS*, a influência populacional do *PRVNS* gerou uma melhora bastante significativa no valor da solução e na convergência do algoritmo como visto em todos os casos da Tabela 2 e da Figura 3. A melhora na solução ocorre pela troca de informação da população no processo de perturbação, o que aumenta a diversidade da busca. A intensificação do algoritmo é feita na etapa de avaliação e troca de solução através da estratégia gulosa, além do controle da amplitude de vizinhança para cada indivíduo de forma independente. O controle de amplitude separada para cada indivíduo permite intensificar ou diversificar de forma independente diferentes regiões do espaço de busca.

Figura 3. Gráfico de convergência com a média da melhor solução por geração ($d = 250$)



6. Conclusão

O presente trabalho apresenta uma abordagem populacional para o algoritmo *VNS*, tendo como foco de aplicação problemas com domínio contínuo. O algoritmo populacional, *PRVNS*, permite usar várias soluções candidatas para explorar o espaço de soluções, em que cada indivíduo pode definir as variações de vizinhança de forma independente através da amplitude. O processo de perturbação usa informação de outras soluções da população influenciadas por um valor de peso que define a amplitude de vizinhança e assim, a sua variação. Dos resultados obtidos, comparando a abordagem populacional proposta (PRVNS) com sua respectiva abordagem não populacional (*RVNS*) o algoritmo *PRVNS* se mostra muito superior ao *RVNS* dada a característica de paralelismo implícito presente na abordagem populacional. Comparando o *PRVNS* com outro algoritmo populacional, o *DE*, os resultados se mostram bastante competitivos e indicando um grande potencial do algoritmo *PRVNS* na resolução de problemas contínuos.

Como trabalhos futuros pretende-se utilizar um conjunto maior de funções objetivo e outros algoritmos populacionais para comparação, além de realizar uma análise dos

parâmetros usados no *PRVNS*. Objetiva-se também aplicar este algoritmo em problemas reais.

Referências

Alba, E. and Martí, R. (2006). *Metaheuristic Procedures for Training Neural Networks*. Operations Research/Computer Science Interfaces Series. Springer.

Carrizosa, E., Martin-Barragan, B., and Romero Morales, D. (2012). Variable neighborhood search for parameter tuning in support vector machines. Technical report, Tech. rep.

Dorigo, M. and Birattari, M. (2010). Ant colony optimization. In *Encyclopedia of Machine Learning*, pages 36–39. Springer.

Gendreau, M. and Potvin, J.-Y. (2010). *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition.

Hsiao, P.-C., Chiang, T.-C., and Fu, L.-C. (2012). A vns-based hyper-heuristic with adaptive computational budget of local search. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8.

Kennedy, J. (2010). Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer.

Kratica, J., Savić, A., Filipović, V., and Milanović, M. (2010). Solving the task assignment problem with a variable neighborhood search. *Serdica Journal of Computing*, 4(4):435p–446p.

Mladenović, N., Dražić, M., Kovačevic-Vujčić, V., and Čangalović, M. (2008). General variable neighborhood search for the continuous optimization. *European Journal of Operational Research*, 191(3):753–770.

Ng, C. (2010). *A Population Based Variable Neighborhood Search Algorithm for the Minimum Shift Design Problem*. Erasmus University.

Pedersen, M. E. H. (2010). Good parameters for differential evolution. *Technical report, Hvass Computer Science Laboratories*.

Price, K., Storn, R. M., and Lampinen, J. A. (2006). *Differential evolution: a practical approach to global optimization*. Springer.

Ribeiro, C. C., Aloise, D., Noronha, T. F., Rocha, C., and Urrutia, S. (2008). An efficient implementation of a vns/ils heuristic for a real-life car sequencing problem. *European Journal of Operational Research*, 191(3):596 – 611.

Sheikh Rajab, R. (2012). Some applications of continuous variable neighbourhood search metaheuristic (mathematical modelling). *School of Information Systems, Computing and Mathematics*.

Wang, X. and Tang, L. (2009). A population-based variable neighborhood search for the single machine total weighted tardiness problem. *Computers & Operations Research*, 36(6):2105–2110.