Algoritmo Populacional de Busca em Vizinhança Variável Aplicado em Otimização Contínua

Wesklei Migliorini
Departamento de Ciência da Computação
Universidade do Estado de Santa Catarina
Joinville - SC - Brasil
Email: wesklei.m@gmail.com

Rafael Stubs Parpinelli
Programa de Pós-graduação em Computação Aplicada
Universidade do Estado de Santa Catarina
Joinville - SC - Brasil
Email: rafael.parpinelli@udesc.br

***Resumo*—Este trabalho apresenta uma abordagem populacional para o algoritmo de Busca em Vizinhança Variável, denominado *PRVNS*. Este trabalho tem como foco de aplicação problemas de otimização com domínio contínuo. Foram utilizadas várias funções *benchmark* com alta dimensionalidade ($D = 250$), a fim de avaliar o desempenho do algoritmo proposto. As principais contribuições do algoritmo *PRVNS* são: evoluir uma população de indivíduos e permitir que cada indivíduo adapte 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. Resultados foram obtidos e comparados com a abordagem não-populacional do *VNS* e com os algoritmos populacionais de Evolução Diferencial, Otimização por Enxame de Partículas e Colônia Artificial de Abelhas. Resultados sugerem que a abordagem proposta é uma alternativa promissora e competitiva para otimização contínua.**

***Keywords*—Otimização Contínua, Algoritmo Populacional, Vizinhança Variável, Meta-heurística.**

I. 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 [1] e sequenciamento de produção [2], quanto em domínios contínuos, como no treinamento de redes neurais [3] e máquina de vetores de suporte [4]. 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 [5]. Um algoritmo populacional possui várias soluções candidatas (indivíduos) que são otimizadas dentro de um processo iterativo e interativo. 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*) [6], Evolução Diferencial (*DE*) [7] e Colônia Artificial de Abelhas (*ABC*) [8].

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 [9] 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 [10] 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 esta não seja melhor, é feito um torneio de dois indivíduos aleatórios da população substituindo o pior. No trabalho de [11] aborda-se o problema *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 pré-estabelecida.

Este artigo propõe uma versão populacional (*PRVNS*) do algoritmo *RVNS* para problemas de otimização contínua utilizando a amplitude de vizinhança no processo de perturbação. 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 autônomo 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 [12]. 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 gerada for melhor que a atual. Quando ocorre a troca de solução, a amplitude é reduzida para intensificar a exploração na vizinhança e quando não ocorre melhora a amplitude é aumentada com objetivo de explorar regiões mais distantes. Em [13] o algoritmo *PRVNS* foi primeiramente apresentado com um conjunto de testes reduzido e com análises preliminares. No trabalho aqui apresentado, um conjunto de testes bem maior é utilizado (23 funções) e uma análise estatística dos resultados obtidos é realizada comparando o algoritmo *PRVNS* com três algoritmos de otimização populacionais bastante difundidos na literatura (*DE*, *PSO* e *ABC*). 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.

II. 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 [14].

Uma estrutura de vizinhança é denotada por $\mathcal{N}_k(\vec{x})$, sendo $\vec{x}$ a solução corrente e k o índice da estrutura de vizinhança sendo explorada ($1 \leq k \leq k_{max}$). 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 $\vec{y}$ próximos a ele que obedecem a estrutura e amplitude de vizinhança utilizadas. Para delimitar os vizinhos em uma região pode-se usar uma amplitude ou raio definido por $r_k(1 \leq k \leq k_{max})$ em que, ao longo do processo de busca, este pode variar, ampliando ou contraindo a amplitude de vizinhos de $\vec{x}$ [15].

Para a estrutura de vizinhança $\mathcal{N}_k(\vec{x})$ é possível utilizar uma ou mais métricas na forma

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

ou ainda

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

onde $\rho_k(\vec{x}, \vec{y})$ é a métrica em uso e caracteriza a distância entre duas soluções dada por

$$\rho_k(\vec{x}, \vec{y}) = (\sum_{i=1}^{d} |x_i - y_i|^p)^{\frac{1}{p}} \qquad (1 \leq p \leq \infty) \quad (3)$$

ou

$$\rho_k(\vec{x}, \vec{y}) = \max_{0 \leq i \leq n} |x_i - y_i|, p = \infty \quad (4)$$

onde p define a métrica ℓ_p: Manhattan (ℓ_1), Euclidiana (ℓ_2) ou Chebyshev (ℓ_∞). Estas métricas são apresentadas na Figura 1 e definem a geometria da estrutura $\mathcal{N}_k(\vec{x})$ no espaço de busca. Sua utilização 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 modelagem do algoritmo [4].

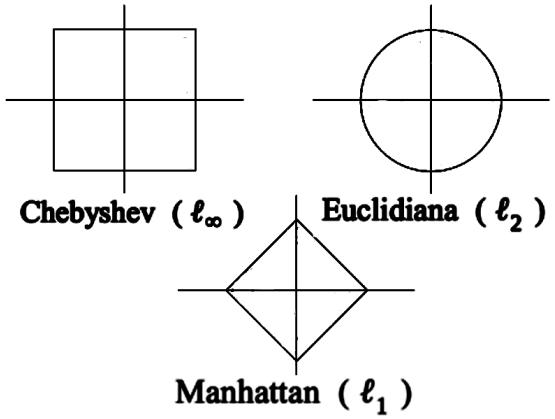

Figura 1: Tipos de métricas . Adaptado de [5]

As estruturas de vizinhança para uma 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 e atribui o valor de $k = 1$ como visto no passo **(b)**. Em **(c)**, k é ampliado até chegar em 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, respectivamente [12].

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 do problema sendo otimizado. 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, o tempo de *CPU*, ou um número máximo de iterações (t_{max}) [3]. O Algoritmo 1 detalha a abordagem *RVNS* [14]. No Algoritmo 1, $\vec{LB}$ e $\vec{UB}$ representam os vetores que restringem o domínio inferior e superior das variáveis do problema, respectivamente.

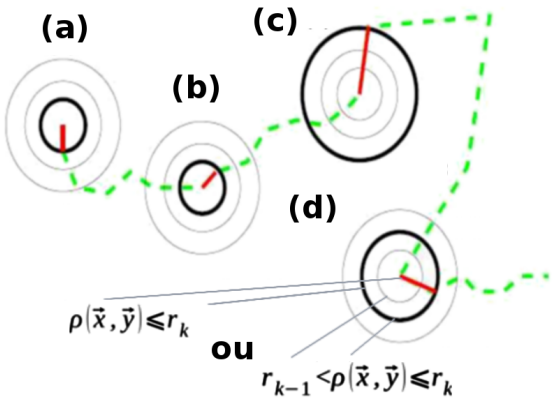

Figura 2: Busca em vizinhança variável com ℓ_2. Adaptado de [12]

Algoritmo 1: Algoritmo RVNS

Entrada: $\mathcal{N}_k, \vec{LB}, \vec{UB}, t_{max}, k_{max}, d$
Saída: Melhor solução encontrada $\vec{x}$

Definir a métrica e estrutura de vizinhança $\mathcal{N}_k$,k = 1, ..., k_{max};
Gerar aleatoriamente um ponto inicial $\vec{x}$;
Avaliar $f(\vec{x})$;
repita
  $k \leftarrow 1$;
  repita
    // Perturba o ponto local gerando um novo candidato ;
    $\vec{y} \leftarrow \vec{x}$;
    $i \leftarrow rand(1, d)$ // índice aleatório ;
    $y[i] \leftarrow rand(LB_i, UB_i) \in \mathcal{N}_k$;
    Avaliar $f(\vec{y})$;
    // Troca a solução e a vizinhança ;
    se $f(\vec{y}) < f(\vec{x})$ // Minimização ;
    então
      $\vec{x} \leftarrow \vec{y}$ // Faz um movimento
      $k \leftarrow 1$ // Volta a primeira amplitude de vizinhança
    senão
      $k++$ // Amplia a vizinhança
  até $k \leq k_{max}$;
  $t++$ // Incrementa o contador de iteração
até $t < t_{max}$;
retorna $\vec{x}$;

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

III. 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.

Algoritmo 2: Esquema geral do algoritmo PRVNS

Entrada: n, PC, G_{max}, k_{max}, r_{k_i} $(1 \leq k_i \leq k_{max})$, $\mathcal{N}_k$, d
Saída: Melhor solução encontrada $\vec{s}$

para *(i=0; $i < n$; i++)* **hacer**
  Inicializar $\vec{x_i}$ aleatoriamente
  Inicializar vizinhança $k_i = 1$
  Avaliar função objetivo $f(\vec{x_i})$
repita
  // Etapa populacional
  para *(i=0; $i < n$; i++)* **hacer**
    // Etapa de perturbação
    Selecionar índices $s_1, s_2 \in n$ aleatoriamente, $s_1 \neq s_2 \neq i$
    Selecionar dimensão $p \leq d$
    para *(j=0; $j < d$; j++)* **hacer**
      se $(j == p) \vee (rand(0,1) \leq PC)$ **então**
        $y_j = x_{s_2,j} + rand(-r_{k_i}, r_{k_i}) * (x_{s_1,j})$
      senão
        $y_j = x_{ij}$
    // Etapa de avaliação e troca de amplitude da vizinhança
    Avaliar $f(\vec{y})$
    se $f(\vec{y}) < f(\vec{x_i})$// Minimização
    então
      $\vec{x_i} \leftarrow \vec{y}$ // Faz um movimento
      $k_i \leftarrow 1$ // Volta a primeira amplitude de vizinhança
    senão
      // Troca de amplitude
      se *($k_i < k_{max}$)* **então**
        k_i++
  $iter++$
até $(iter < G_{max})$;
$\vec{s} \leftarrow$ Encontrar melhor solução da população ;
retorna $\vec{s}$;

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}, o valor de amplitude atribuído a cada vizinhança r_{k_i}, a estrutura $\mathcal{N}_k$ a ser utilizada e a dimensão da função otimizada d. No laço da linha 1, 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 5 até a condição de parada ser alcançada. Neste laço estão as principais etapas do *PRVNS*: perturbação e troca da amplitude de vizinhança. A perturbação ocorre entre as linhas 7 e 13. No *RVNS* a perturbação é feita modificando um índice de $\vec{x}$ para gerar $\vec{y}$. No *PRVNS* é utilizada a informação da solução de outros indivíduos da população para compor o vetor modificado (linha 7). O valor de p selecionado na linha 8 garante que pelo menos um índice de x_i seja perturbado, porém é mantido o parâmetro PC para controlar a probabilidade de perturbação para os demais índices. O controle de expansão e contração faz analogia com a amplitude de vizinhança do *VNS* e é visto na linha 7. Nela, dois indivíduos aleatórios são selecionados e usados na linha 11 junto ao valor de peso $rand(-r_{k_i}, r_{k_i})$, que controla a amplitude de vizinhança. Este valor possui característica estocástica no intervalo de raio atual de vizinhança, o que leva expansão e contração ao modificar o raio iterativamente. A nova coordenada y_j da linha 11 pode estar aproximada ou distanciada de $x_{s_2,j}$ e $x_{s_1,j}$, de acordo com o valor positivo ou negativo do peso. Para isto é feita a soma do índice j de x_{s_2} ao de x_{s_1} com o peso aplicado nesta mesma linha. Entre as linhas 14 e 21 é feita a troca de vizinhança. Na linha 14 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 17 e 18. 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 21 com o objetivo de explorar regiões mais distantes no espaço de busca. A amplitude é incrementada até que o valor de k_i alcance $k_{i_{max}}$ (linha 21 e 22) seja alcançado. O algoritmo encerra com a condição de parada G_{max} que é o número máximo de gerações.

IV. Experimentos

Neste trabalho, o algoritmo *PRVNS* é comparado com outros três algoritmos populacionais amplamente usados na literatura, Evolução Diferencial (*DE*), Otimização por Enxame de Partículas (*PSO*) e Colônia Artificial de Abelhas (*ABC*), e com o algoritmo canônico *RVNS*.

O *DE* é um algoritmo populacional que utiliza um peso fixo F para perturbação fazendo a diferença entre três indivíduos selecionados aleatoriamente [7]. O algoritmo *ABC* é inspirado no comportamento das abelhas na busca de alimentos [8] e o algoritmo *PSO* é inspirado no comportamento coordenado do movimento de cardumes de peixes e revoada de pássaros [6].

Os algoritmos foram aplicados a um conjunto de 23 funções *benchmark* para otimização contínua, descritas na Tabela I. A tabela mostra a sigla de cada função usada no trabalho, bem como o nome, 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, ambiente *Linux 64 bits* e linguagem de programação C. Para todos os algoritmos foram utilizadas 500.000 avaliações da função objetivo em cada execução. Os algoritmos *PRVNS*, *DE*, *PSO* e *ABC* 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 dimensionalidade 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 recomendados pela literatura[1] para $PC = 0,9$ e $F = 0,4717$ [16]. O $PC = 0,9$ também foi utilizado para o *PRVNS*. O *PSO* também usa os valores da literatura[2] com $W = 0,721$ e $\varphi_p = \varphi_g = 1,193$. O algoritmo *ABC*[3] utiliza *limit=100* [8].

Tabela I: Funções avaliadas no experimento

	Função	Domínio	Min.
F1	Rastrigin	$[-5,12,5,12]^d$	0
F2	Schaffer F7	$[-100,100]^d$	0
F3	Ackley	$[-32,32]^d$	0
F4	Rosenbrock	$[-30,30]^d$	0
F5	Sphere	$[-100,100]^d$	0
F6	Schaffer F6	$[-100,100]^d$	0
F7	Levy	$[-10,10]^d$	0
F8	Zakharov	$[-5,10]^d$	0
F9	Schwefel 2,22	$[-10,10]^d$	0
F10	Griewank	$[-600,600]^d$	0
F11	M. Potential Energy	$[0,5]^d$	$-0,0411183034n$
F12	G. Schwefels 2,26	$[-500,500]^d$	$-418,982887272433$
F13	Step	$[-100,100]^d$	0
F14	G. Penalized f1	$[-50,50]^d$	1,57e-032
F15	Egg Holder	$[-512,512]^d$	$-915,61991n + 862,10466$
F16	Generalized Holzman	$[-10,10]^d$	0
F17	Michalewitz	$[0,\pi]^d$	$-0,99864n + 0,30271$
F18	G. Penalized f2	$[-50,50]^d$	1,34e-032
F19	Powell	$[-4,5]^d$	0
F20	Rana	$[-512,512]^d$	$-511,70430n + 511,68714$
F21	Shubert	$[-10,10]^d$	-24,06
F22	StretchedV	$[-10,10]^d$	0
F23	Multimod	$[-10,10]^d$	0

V. Resultados e Análises

Os resultados dos experimentos para as vinte e três funções *benchmark* são mostrados na Tabela II com o número de cada função e a média com desvio padrão para os algoritmos *RVNS*, *PRVNS*, *DE*, *PSO* e *ABC*, respectivamente. Em negrito está ressaltado o algoritmo que obteve o melhor resultado para determinada função levando em consideração a análise estatística. Na parte inferior da tabela, a quantidade de melhores soluções e de soluções ótimas encontradas foram sumarizadas para cada algoritmo.

Na Tabela II o *PRVNS* foi o único algoritmo a encontrar a solução ótima (quarta coluna da Tabela I) em F3, F5, F9, F16 e F23 além de obter 12 melhores soluções em F2, F3, F4, F5, F9, F13, F14, F16, F18, F19, F22 e F23 seguido do *ABC* com 6 melhores em F6, F12, F15, F17, F20 e F21. O *DE* obteve 5 melhores soluções em F1, F7, F8 e F10 e é

[1]DE C Code: http://www1.icsi.berkeley.edu/ storn/code.html
[2]Standard PSO (PSO-07): http://www.particleswarm.info/Programs.html
[3]ABC C Code: http://mf.erciyes.edu.tr/abc/

estatisticamente equivalente ao *ABC* em F6. Já o *PSO* foi melhor apenas em F11 e é estatisticamente equivalente ao *ABC* e F12. Por último, o *RVNS* não obteve a melhor solução em nenhum caso. Nenhum dos algoritmos além do *PRVNS* obteve solução ótima durante os experimentos.

Tabela II: Resultados obtidos para os experimentos com as funções ($d = 250$)

Número		RVNS	PRVNS	DE	PSO	ABC
F1	Média	4,60e+03	1,92e+03	**264,60**	8,98e+02	1,48e+03
	Desv. P.	1,53e+02	2,68e+02	**32,64**	2,36e+02	5,87e+01
F2	Média	45,82	**3,97**	15,91	21,60	29,40
	Desv. P.	9,65	**1,43**	0,59	0,54	0,59
F3	Média	21,21	**0,00**	8,62	8,62	18,30
	Desv. P.	0,06	**0,00**	0,63	0,78	0,33
F4	Média	4,10e+09	**247,00**	4,65e+03	1,12e+06	3,10e+08
	Desv. P.	3,47e+08	**0,81**	6,62e+03	3,98e+05	7,21e+07
F5	Média	8,38e+05	**0,00**	23,36	3,51e+03	1,35e+05
	Desv. P.	5,17e+04	**0,00**	118,60	9,56e+02	1,44e+04
F6	Média	122,52	114,00	**109,00**	116,00	**105,00**
	Desv. P.	2,18	0,88	**4,88**	1,03	**1,76**
F7	Média	3,29e+03	1,02e+03	**34,27**	62,70	353,00
	Desv. P.	2,70e+02	1,38e+02	**4,91**	10,60	47,80
F8	Média	2,16e+18	1530,00	**1234,27**	1,05e+05	4,13e+03
	Desv. P.	9,51e+17	226,00	**141,28**	5,37e+05	1,63e+02
F9	Média	1180,00	**0,00**	0,07	46,80	362,00
	Desv. P.	35,60	**0,00**	0,17	7,32	19,60
F10	Média	7,05e+03	1,00	**0,06**	35,10	1,18e+03
	Desv. P.	3,26e+02	0,00	**0,28**	14,30	1,63e+02
F11	Média	241,00	121,00	71,60	**14,90**	45,60
	Desv. P.	9,70	10,10	45,40	**5,23**	3,47
F12	Média	-1,96e+03	-3,75e+04	-1,46e+04	**-4,90e+04**	**-5,73e+04**
	Desv. P.	2,71e+03	3,38e+03	1,68e+03	**7,81e+03**	**1,76e+03**
F13	Média	7,84e+05	**32,10**	6,64e+03	3,64e+03	1,30e+05
	Desv. P.	3,17e+04	**1,81**	1,28e+03	1,36e+03	1,34e+04
F14	Média	2,68e+14	**1,17e+06**	1,63e+11	5,95e+10	2,05e+13
	Desv. P.	1,34e+13	**5,31e+05**	6,03e+10	3,20e+10	5,02e+12
F15	Média	-5,49e+03	-5,08e+04	-1,86e+04	-5,52e+04	**-9,08e+04**
	Desv. P.	4,90e+03	4,21e+03	1,36e+03	1,68e+04	**3,15e+03**
F16	Média	6,17e+07	**0,00**	4,63e+04	1,28e+04	4,00e+06
	Desv. P.	6,05e+06	**0,00**	2,38e+04	3,56e+03	1,45e+06
F17	Média	-41,30	-71,90	-56,60	-61,20	**-119,00**
	Desv. P.	3,17	6,25	1,67	3,25	**3,07**
F18	Média	1,70e+10	**6,52e+04**	4,31e+06	1,01e+06	1,26e+09
	Desv. P.	1,37e+09	**5,93e+04**	2,97e+06	7,89e+05	3,96e+08
F19	Média	4,69e+05	**0,09**	457,00	273,00	1,63e+04
	Desv. P.	8,98e+04	**0,05**	156,00	93,20	5,42e+03
F20	Média	-14,80	-125,00	-50,80	-164,00	**-252,00**
	Desv. P.	14,00	12,60	3,22	47,60	**5,16**
F21	Média	-1,42	-7,10	-3,13	-5,13	**-12,60**
	Desv. P.	0,61	0,83	0,28	0,43	**0,33**
F22	Média	956,00	**2,85**	274,00	389,00	580,00
	Desv. P.	24,5	**3,91**	14,10	18,90	14,20
F23	Média	2,20e+87	**0,00**	59,10	56,00	50,80
	Desv. P.	1,18e+88	**0,00**	8,83	11,00	6,65
Melhores soluções:		**0**	**12**	**5**	**2**	**6**
Ótimo global:		**0**	**5**	**0**	**0**	**0**

Dois testes estatísticos não paramétricos foram aplicados: *Friedman Rank Sum Test* e *Wilcoxon Signed Rank Test* [17]. Optou-se por testes não-paramétricos dada a não-normalidade das variáveis envolvidas verificada com a aplicação do teste *Shapiro-Wilk* (*p-value=0,05*). O teste com *Friedman* detecta se há diferenças significativas entre o valor obtido pelos algoritmos, podendo determinar se o valor das amostras selecionadas ao acaso são realmente aleatórias e diferentes. Por esta técnica, quanto menor o ranqueamento obtido melhor o resultado. Para a Tabela II, os valores obtidos aplicando o teste de *Friedman* são: *PRVNS* $= 1,91$, *DE* $= 2,67$, *PSO* $= 2,54$, *ABC* $= 2,95$ e *RVNS* $= 5$. Através deste teste, o *PRVNS* se mostra em posição melhor em relação aos outros algoritmos.

Como o *Friedman Test* faz a análise de todos os algoritmos, não é possível tirar conclusões mais precisas de comparação pareada, ou seja, comparar o melhor entre cada par de algoritmos. Por isso o *Wilcoxon Test* é utilizado como complemento. O teste com *Wilcoxon* compara os resultados de dois algoritmos buscando identificar se existe diferenças com significância estatística entre eles. Para analisar estatisticamente que um algoritmo é melhor do que outro, é verificada a hipótese nula em que os resultados são equivalentes e espera-se rejeitar esta hipótese. Ao rejeitar a hipótese nula, considerando uma significância estatística, tem-se como consequência que um algoritmo é melhor que outro. No teste de *Wilcoxon* realizado para cada função utiliza-se o valor das 30 execuções e pareia-se os resultados obtidos pelos algoritmos *RVNS*, *DE*, *PSO* e *ABC* com os resultados obtidos pela abordagem *PRVNS*. O teste de hipótese nula usa *p-value* igual a $0,05$

Dos resultados obtidos com o teste de *Wilcoxon*, apenas o pareamento *PRVNS* e *PSO* obteve *p-value* maior que *0,05* na função F15, confirmando a hipótese nula de que os resultados obtidos por estes algoritmos para esta função são estatisticamente equivalentes. Para todos os outros pareamentos e demais funções, os *p-values* obtidos foram menores que *0,05*, rejeitando a hipótese nula indicando que os resultados obtidos nos experimentos são estatisticamente diferentes.

Comparando o *PRVNS* com o *RVNS*, a influência populacional do *PRVNS* gerou uma melhora bastante significativa no valor da solução final obtida como visto em todos os casos da Tabela II e nos testes estatísticos aplicados. A melhora na solução ocorre pela troca de informações 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 separado para cada indivíduo permite intensificar ou diversificar em diferentes regiões do espaço de busca.

VI. 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 outros algoritmos populacionais, *DE*, *PSO* e *ABC*, os resultados se mostram bastante competitivos indicando um grande potencial do algoritmo *PRVNS* na resolução de problemas contínuos.

Como trabalhos futuros pretende-se realizar uma análise dos parâmetros usados no *PRVNS* e aplicar o algoritmo em problemas do mundo real.

Agradecimentos: Os autores gostariam de agradecer à Universidade do Estado de Santa Catarina (UDESC) pelo apoio financeiro.

Referências

[1] J. Kratica, A. Savić, V. Filipović, and M. Milanović, "Solving the task assignment problem with a variable neighborhood search," *Serdica Journal of Computing*, vol. 4, no. 4, pp. 435p–446p, 2010.

[2] C. C. Ribeiro, D. Aloise, T. F. Noronha, C. Rocha, and S. Urrutia, "An efficient implementation of a vns/ils heuristic for a real-life car sequencing problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 596 – 611, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221707001191

[3] E. Alba and R. Martí, *Metaheuristic Procedures for Training Neural Networks*, ser. Operations Research/Computer Science Interfaces Series. Springer, 2006. [Online]. Available: http://books.google.com.br/books?id=6X-drhJaxvsC

[4] E. Carrizosa, B. Martin-Barragan, and D. Romero Morales, "Variable neighborhood search for parameter tuning in support vector machines," Tech. rep, Tech. Rep., 2012.

[5] P. Hansen and N. Mladenovic, "Variable neighborhood search," in *Handbook of Metaheuristics*, ser. International Series in Operations Research and Management Science, F. Glover and G. Kochenberger, Eds. Springer US, 2003, vol. 57, pp. 145–184. [Online]. Available: http://dx.doi.org/10.1007/0-306-48056-5_6

[6] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 760–766.

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

[8] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[9] C. Ng, *A Population Based Variable Neighborhood Search Algorithm for the Minimum Shift Design Problem*. Erasmus University, 2010. [Online]. Available: http://books.google.com.br/books?id=_JKhkQEACAAJ

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

[11] X. Wang and L. Tang, "A population-based variable neighborhood search for the single machine total weighted tardiness problem," *Computers & Operations Research*, vol. 36, no. 6, pp. 2105–2110, 2009.

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

[13] W. Migliorini and R. S. Parpinelli, "Uma abordagem populacional para o algoritmo de busca em vizinhanca variavel aplicado em otimizacao continua," *Computer on the Beach*, vol. 1, pp. 11–20, 2015.

[14] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics*, 2nd ed. Springer Publishing Company, Incorporated, 2010.

[15] N. Mladenović, M. Dražić, V. Kovačevic-Vujčić, and M. Čangalović, "General variable neighborhood search for the continuous optimization," *European Journal of Operational Research*, vol. 191, no. 3, pp. 753–770, 2008.

[16] M. E. H. Pedersen, "Good parameters for differential evolution," *Technical report, Hvass Computer Science Laboratories*, 2010.

[17] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability," *Soft Computing*, vol. 13, no. 10, pp. 959–977, 2009.