UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO DE ENGENHARIA DE PRODUÇÃO

Estratégias de Aplicações Sequenciais e Paralelas da Metaheurística Otimização por Enxame de Partículas ao Problema do Caixeiro Viajante

por

Thales Lima Silva

Orientador

Prof. D. Sc. Dario José Aloise

Natal, RN
2008

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

THALES LIMA SILVA

Estratégias de Aplicações Seqüenciais e Paralelas da Metaheurística Otimização por Enxame de Partículas ao Problema do Caixeiro Viajante

Dissertação de Mestrado apresentada à Universidade Federal do Rio Grande do Norte como parte dos requisitos para obtenção do Título de Mestre em Engenharia de Produção.

Orientador:
Prof. Dr. Dario José Aloise

Natal,RN
2008

AGRADECIMENTOS

A minha família, pelo apoio aos estudos, carinho e ensinamentos de vida.

A minha esposa, Roberta, pela compreensão nos momentos de ausência para dedicação a este trabalho e apoio nos momentos mais difíceis.

A toda equipe da SIN Tecnologia, pela paciência no período das disciplinas.

Ao professor Dario Aloise, pela orientação fornecida desde os tempos da graduação, pela paciência e pelas idéias que enriqueceram este trabalho.

Ao grupo de estudos do PROMETH, em especial a Pedro Maia, que me tirou várias dúvidas sobre o MPI, as comunidades relacionadas ao estudo do PSO na internet, onde trocamos várias idéias sobre esta técnica.

A meus amigos de infância que apesar de todos estarem longe fisicamente, mas sempre mantemos contatos e apoiamos uns aos outros em todas as batalhas, formando uma grande família.

A todos os mestres, das tias do jardim de infância aos doutores da UFRN, pois foram e são de fundamental importância para minha formação intelectual e profissional.

Enfim a Deus, que nos guia nos momentos mais difíceis e nos acompanha nos momentos de alegria e a duas "*estrelas"* em especial, Luisa Xabregas, minha afilhada, uma estrelinha que veio ao mundo para brilhar e trazer muita alegria e felicidade a todos; e a Cícero Bringel, meu sogro, uma estrela que hoje brilha no céu e nos deixou muitos ensinamentos e saudades.

RESUMO

Otimização por Enxame de Partículas ou *Particle Swarm Optimization* (PSO) é uma metaheurística que surgiu na intenção de simular o comportamento de um conjunto de pássaros em vôo, com seu movimento localmente aleatório, mas globalmente determinado. Esta técnica tem sido muito utilizada na resolução de problemas contínuos não-lineares e ainda pouco explorada em problemas discretos. Este trabalho apresenta o funcionamento desta metaheurística, além de propor estratégias para sua aplicação em problemas de otimização discreta tanto na sua forma de execução seqüencial quanto paralela. Os experimentos computacionais foram realizados para instâncias do problema do caixeiro viajante, selecionados na biblioteca TSPLIB contendo até 1002 nós, mostrando a melhoria de desempenho dos métodos paralelos em relação as suas versões seqüenciais, em tempo de execução e resultados.

Palavras-chave: Enxame de Partículas, Problema do Caixeiro Viajante, Metaheurística, Paralelismo, .

Abstract

Particle Swarm Optimization is a metaheuristic that arose in order to simulate the behavior of a number of birds in flight, with its random movement locally, but globally determined. This technique has been widely used to address non-liner continuous problems and yet little explored in discrete problems. This paper presents the operation of this metaheuristic, and propose strategies for implementation of optimization discret problems as form of execution parallel as sequential. The computational experiments were performed to instances of the TSP, selected in the library TSPLIB contenct to 3038 nodes, showing the improvement of performance of parallel methods for their sequential versions, in executation time and results.

Keywords: *Particle Swarm*, Traveling Salesman Problem, Metaheuristic, Parallel.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AS – Ant System

ACO – Ant Colony Optimization

EA – Evolutionary Algorithms

ES – Evolutionary Strategy

GA – Genetic Algorithm

GRASP – Greedy Randomized Adaptative Search Procedure

LCR – Lista de Candidatos Restrita

NN – Neural Networks

NP – Nondeterministic Polynomial Time

PSO – Particle Swarm Optimization

SA – Simulated Annealing

TS – Tabu Search

TSP – Traveling Salesman Problem

TSPLIB – Traveling Salesman Problem Library

VNS – Variable Neighborhood Search

MPI – Message Passing Interface

Capítulo 1

Considerações Iniciais

1.1 Introdução

Nas últimas décadas com a "popularização" da computação, cresceu também a necessidade da aplicação da mesma, na resolução de diversos tipos de problemas do mundo real, partindo desde sua formalização até a resolução dos mesmos, através da implementação de algoritmos eficientes, capazes de solucioná-los. Entre estes problemas destacam-se os de otimização combinatória NP - difíceis (RIBEIRO, 1998), que são problemas de otimização nos quais não existem algoritmos polinomiais que os resolva, a menos que P=NP. Para resolução destes problemas, ocorreu um aumento no estudo das heurísticas (RIBEIRO, 1998), que são métodos aproximados projetados com base nas propriedades estruturais ou nas características das soluções dos problemas, com complexidade reduzida (polinomial) em relação a dos algoritmos exatos e fornecendo, em geral, soluções viáveis de boa qualidade próximas da solução ótima. Entre estes temos: os métodos construtivos, os de busca local e as metaheurísticas, sendo esta última área onde este trabalho esta focado.

No estudo das metaheurísticas, podemos citar o surgimento do conceito de evolução, onde a primeira utilização para encontrar soluções aproximadas em problemas NP - difíceis, fora utilizada por Rechenberg e Schwefel nos princípios dos anos sessenta e foi denominada de Estratégia Evolucionária (ES), uma técnica simples e com baixas exigências computacionais (RECHENBERG, 1981). Sua base ocorre através da observação da natureza, sendo possível a formulação de vários problemas de difícil solução, tais como o modo que os insetos encontram o caminho para um local que tenha comida, como os predadores capturam suas presas ou como os pássaros definem um melhor lugar para pousar, entre outros. Parte destes problemas é resolvida através de aprendizado e

adaptação, mas a maioria deles é solucionada através da evolução, baseada nos princípios de Darwin (RIBEIRO, 1996).

Com o crescimento do poder computacional, as técnicas de busca que anteriormente eram bastante simples, tornaram-se mais elaboradas e hoje existe um foco maior nas buscas baseadas em população. Citando como exemplo, temos os Algoritmos Evolucionários (EA) e o algoritmo baseado em Colônia de Formigas (AS) que hoje são utilizados em uma gama de problemas, tais como otimização numérica, combinatória, bioinformática, problemas de alocação, entre outros.

Seguindo com o estudo em técnicas evolucionárias, em 1995, Kennedy e Eberhart (KENNEDY;EBERHART, 2001) apresentaram uma técnica chamada Otimização por Enxame de Partículas (PSO - Particle Swarm Optimization), que é baseada na simulação de um sistema social simplificado, tendo sua intenção original de simular computacionalmente, a coreografia realizada por bando de pássaros.

Em sua versão original, o PSO é muito similar a algumas técnicas populacionais, como os Algoritmos Genéticos, sistema é inicializado com uma população de soluções randômicas, diferenciando-se pelo motivo de nenhum operador inspirado pelos procedimentos de DNA ser aplicado na população (nomeada enxame ou nuvem) para extrair uma nova geração de candidatos a soluções. Ao invés da mutação são utilizados pontos de busca, chamados de partículas, movendo-se em um espaço de busca n-dimensional, sendo cada uma delas uma solução potencial para o problema.

O PSO foi primeiramente desenvolvido para aplicações em otimizações de valores contínuos, mas, recentemente, foi generalizado para incluir espaços de busca discretos (CLERC, 2000). Este trabalho tem como objetivo estudar a técnica de Otimização por Enxame de Partículas e testar sua eficiência em aplicações discretas tanto para execuções seqüências quanto para execuções paralelas,

aplicado ao Problema do Caixeiro Viajante (TSP), já que o mesmo é um problema NP - difícil e normalmente é utilizando como *benchmark* na introdução de novas técnicas na literatura. Portanto, como contribuições temos: desenvolvimento de novos métodos para aplicação do PSO ao TSP, aplicação do mesmo em ambiente paralelo e uma melhoria contínua dos resultados obtidos, tanto no tempo de execução quanto nos resultados obtidos.

Este trabalho está organizado da seguinte forma: o capítulo 2 aborda o Problema do Caixeiro Viajante. O capítulo 3 é dedicado à Computação Evolucionária apresentando suas principais linhas que são os algoritmos evolucionários, algoritmos genéticos e os enxames ou nuvens (*swarms)* inteligentes. No capítulo 4 é feita uma descrição da Otimização por Enxame de Partículas e sua aplicação ao Problema do Caixeiro Viajante. O capítulo 5 apresenta os resultados computacionais obtidos com a implementação dos algoritmos desenvolvidos. Finalmente o capítulo 6 é dedicado as conclusões e sugestões para trabalhos futuros.

Capítulo 2

O Problema do Caixeiro Viajante

2.1 Definição

O Problema do Caixeiro Viajante (PCV) é um dos problemas de otimização combinatória mais estudado no mundo dentre um amplo conjunto de problemas de otimização, seu estudo atraí pesquisadores de diferentes campos, dentre estes estão: pesquisa operacional, matemática, física, biologia, inteligência artificial, entre outros. Isso se deve ao fato de que, apesar da simplicidade de sua formulação, o mesmo não é de resolução tão fácil. Consequentemente, tem sido usado como *benchmark* para avaliação de novos algoritmos e estratégias avançadas de solução de problemas de otimização combinatória, como Busca Tabu (RIBEIRO, 1998), Algoritmos Genéticos (GUEDES; LEITE; ALOISE, 2004), Simulated Annealing (SA) (RIBEIRO, 1998), Redes Neurais (NN), PSO entre outros.

O PCV é composto por um conjunto de *n* cidades, tendo o seguinte objetivo: o caixeiro viajante deve sair de uma cidade chamada origem, visitar cada uma das *n - 1* cidades restantes, apenas uma única vez e retornar à cidade origem percorrendo a menor distância possível. Em outras palavras, deve ser encontrada uma rota fechada (ciclo hamiltoniano) de custo (comprimento) mínimo que passe exatamente uma única vez por cada cidade (RIBEIRO, 1998). As Figuras 1 e 2 mostram, uma possível solução, dentre as várias existentes para um determinado problema.

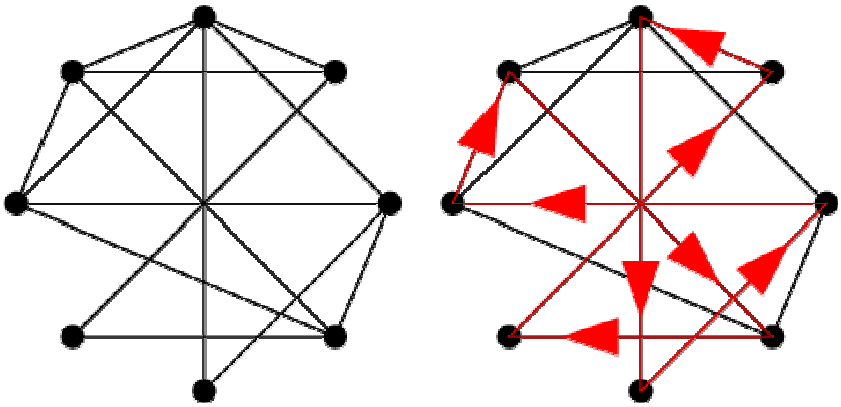

Figura 1: Possíveis Rotas em um Grafo

À medida que o número de cidades vai crescendo o número de **possíveis** soluções, ou seja, de rotas que sejam **possíveis** de se visitar cresce em *O*(N!).

O Problema do Caixeiro Viajante é considerado simétrico, quando dado um conjunto de *n* nós, cada um representando uma cidade do problema, a distância do nó *i* para o nó *j* possui o mesmo valor que da distância do nó *j* para o nó *i*. Caso contrário será considerado assimétrico.

Embora seja difícil encontrar a(s) solução(ões) ótima(s) para o PCV quando o número de cidades é muito grande, com o uso de heurísticas é possível encontrar soluções viáveis muito próximas do ótimo sem que seja necessário enumerar todas as soluções possíveis.

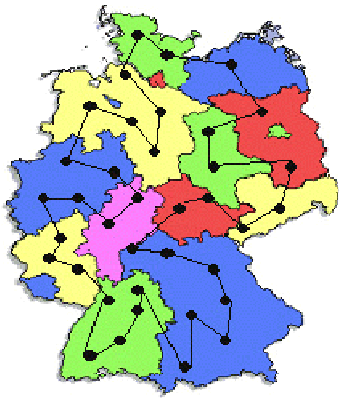

Figura 2: Possível Solução para o Problema do Caixeiro Viajante

2.2 Formulação Matemática

A primeira formulação para o PCV foi apresentada por Dantzig, Fulkerson e Johnson em 1954 (DANTZIG; FULKERSON; JOHNSON, 1954), onde definiram uma variável binária x_{ij} ($i \neq j$) igual a 1 se e somente se o arco (v_i, v_j) seja usado na solução ótima e 0 caso contrário, definido em um grafo G(V, A), onde V é um conjunto de vértices e A um conjunto de arestas.

Formulação:

$$Minimizar \quad z = \sum_{i \neq j} c_{ij} x_{ij} \qquad (2.1)$$

$$Sujeito\ à: \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1,...,n) \qquad (2.2)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1,...,n) \qquad (2.3)$$

$$\sum_{v_i, v_j \in S} x_{ij} \leq |S| - 1, \quad (S \subseteq V;\ 2 \leq |S| \leq n-1) \qquad (2.4)$$

$$x_{ij} \in \{0, 1\} \qquad (i, j = 1,...,n;\ i \neq j) \qquad (2.5)$$

As restrições (2.2) e (2.3) garantem que para cada vértice há somente um arco de entrada e um de saída. A restrição (2.4) garante a eliminação de sub-rotas, porém o número de inequações (2.4) é de ordem exponencial.

Além da questão do desempenho computacional outro ponto de interesse no PCV são as inúmeras aplicações práticas como, por exemplo, em linhas de montagem de placas eletrônicas, onde se busca encontrar a mínima distância para um equipamento de soldar atuar nos pontos desejados na placa (REINELT, 1994), na maioria dos problemas de roteamento de veículos (BODIN;ASSAD;BALL, 1983), otimização do movimento de ferramentas de corte(CHAUNY;HAURIE;WAGNEUR;LOULOU, 1987), na solução de problemas de seqüenciamento(WHITLEY; STARKWHETER; SHANER, 1991), entre outros.

2.3 Métodos de Resolução

Para resolução deste problema, NP - árduo, são utilizados algoritmos exatos, tais como programação dinâmica (COTTA, ALDANA, NEBRO, TROYA, 1995) e branch-and-bound (BERTSKAS, 1987), entre outros. Já para problemas práticos onde o tempo computacional dos algoritmos exatos torna-se inviável utilizam-se métodos heurísticos.

O sucesso de um método heurístico, depende de sua capacidade de:

- Adaptação a instâncias especiais;
- Escapar de ótimos locais;
- Fazer uso da estrutura do problema;
- Estrutura eficiente de dados;
- Pré-processamento;
- Boas técnicas para construir soluções iniciais;
- Melhoria de solução através de busca local;
- Randomização controlada;
- Diversificação de busca quando nenhuma melhoria adicional parece possível;
- Intensificação da busca em regiões promissoras.

Na tentativa de conjugar todas essas características descritas acima, muitos trabalhos foram desenvolvidos nas últimas décadas com o sentido de melhorar os métodos heurísticos, sem, no entanto prejudicar a sua principal característica, que é a flexibilidade. Estes trabalhos deram origem às estratégias comumente conhecidas como metaheurísticas, que são novas classes de heurísticas que melhoram a performance de algoritmos aproximados na resolução do problema.

2.4 Metaheurísticas

Contrariamente às heurísticas convencionais, as metaheurísticas possuem um caráter geral de ter condições de escapar de ótimos locais. Por exemplo, em um problema típico de minimização, encontra-se um ótimo local quando qualquer movimento a ser feito piore o valor atual da função objetivo. Um ótimo global corresponde ao menor valor da função objetivo, dentre todos os ótimos locais existentes no espaço de busca.

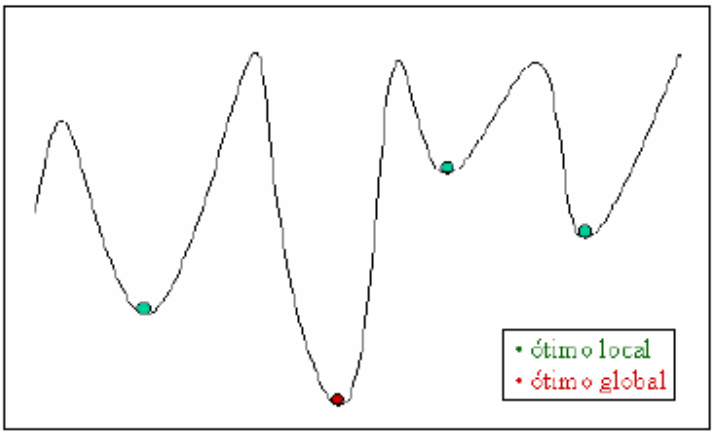

Figura 3: Representação Gráfica para um Problema de Minimização

As metaheurísticas, assim como os métodos de busca local tradicionais, diferenciam-se entre si basicamente pelas seguintes características: critério de escolha de uma solução inicial; definição da vizinhança N(s) de uma solução S; critério de seleção de uma solução vizinha dentro de N(s), além do critério de parada.

Dentre os procedimentos enquadrados como metaheurísticas destacam-se: *Simulated Annealing* (RIBEIRO, 1998), Busca Tabu (TS) (RIBEIRO, 1998) e *Greedy Randomized Adaptative Search Procedure* (GRASP) (RIBEIRO, 1998).

2.4.1. SIMULATED ANNEALING

O Simulated annealing (recozimento simulado), é um método estocástico no qual é caracterizado pela exploração da vizinhança aleatoriamente, o que significa, essencialmente, que o vizinho em direção ao qual se vai deslocar, é escolhido ao acaso na vizinhança da solução corrente.

O Simulated annealing (AARTS; KORST, 1989) é baseado nos resultados da termodinâmica. O primeiro lançamento sobre este assunto nós devemos a (KIRKPATRICK; GELLAT; VECCHI, 1983). O seu princípio está baseado na observação física do comportamento da matéria. Quando um cristal é levado a sua temperatura de fusão, as moléculas estão em um estado muito desordenado e se agitam livremente, ver figura 4.

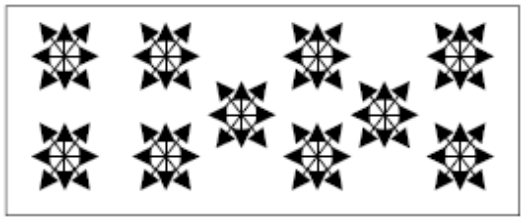

Figura 4: Moléculas em Movimento Desordenado

Resfriando-se bruscamente, o nível de energia baixa rapidamente e as moléculas continuarão em um estado ainda muito desordenado no qual o nível de energia é muito superior ao do cristal perfeito. Por outro lado, resfriando-se a amostra de maneira infinitamente lenta, as moléculas vão adquirir a estrutura cristalina estável que tem um nível de energia mais fraca possível, vide figura 5.

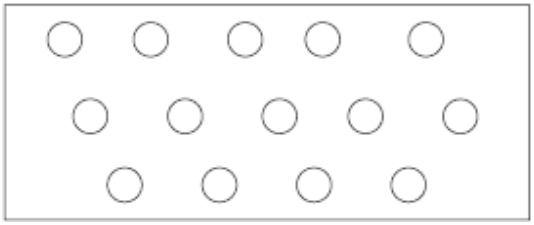

Figura 5: Moléculas com Estrutura Estável

Esta técnica começa sua busca a partir de uma solução inicial qualquer. O procedimento principal consiste em um *loop* que gera aleatoriamente, em cada iteração, um único vizinho S' da solução corrente S.

Considerando um problema de minimização, seja Δ a variação de valor da função objetivo ao mover-se para uma solução vizinha candidata, isto é, $\Delta = f(s') - f(s)$. O método aceita o movimento e a solução vizinha passa a ser a nova solução corrente se $\Delta < 0$. Caso $\Delta \geq 0$, a solução vizinha candidata também poderá ser aceita, mas neste caso, com uma probabilidade $e^{-\Delta/T}$, onde T é um parâmetro do método, chamado de temperatura e que regula a probabilidade de se aceitar soluções de pior custo. Tendo como parâmetros de controle do procedimento a razão de resfriamento α e o número de iterações para cada temperatura (SAmax) e a temperatura inicial T_0.

A seguir o procedimento Simulated Annealing:

```
Procedimento SA (f(.), N(.), α, SA Max, T0, s)
1       s*←s;
2       IterT ←0;
3       T ←T0;
4       enquanto (T > 0) faça
5               enquanto (IterT < SAmax) faça
6                       IterT = IterT +1;
7                       Gere um vizinho qualquer s' ∈ N(s);
8                       Δ = f(s') – f(s);
9                       se Δ < 0
10                              então
11                                      s ← s';
12                                      se s' < s* então s* ← s';
13                              senão
```

```
14                                  Tome X ∈ [0,1]
15                                  se ( x < e^(-Δ/T) ) então s ← s';
16                      fim-se
17              fim-enquanto
18              T ← α *T;
19              IterT = 0;
20 fim-enquanto
21 s ← s*;
22 Retorne S
23 fim SA
```

2.4.2. BUSCA TABU

O método da Busca Tabu foi proposto inicialmente por (GLOVER, 1996), é uma metaheurística de melhoramento local que utiliza uma lista de movimentos proibidos para avançar em direção ao local ótimo. Esta técnica utiliza uma estrutura de memória, chamada Lista Tabu, para guiar um método de descida a continuar a exploração do espaço de soluções mesmo na ausência de movimentos de melhora, evitando que haja a formação de ciclos, isto é, o retorno a um ótimo local previamente visitado.

Mais especificadamente, começando com uma solução inicial S_0, um algoritmo BT explora, a cada iteração, um subconjunto V da vizinhança N(S) da solução corrente S. O membro S' de V com menor valor nessa região segundo a função $f(.)$ torna-se a nova solução corrente mesmo que S' seja pior que S, isto é, que $f(s') > f(s)$, em caso de um problema de minimização.

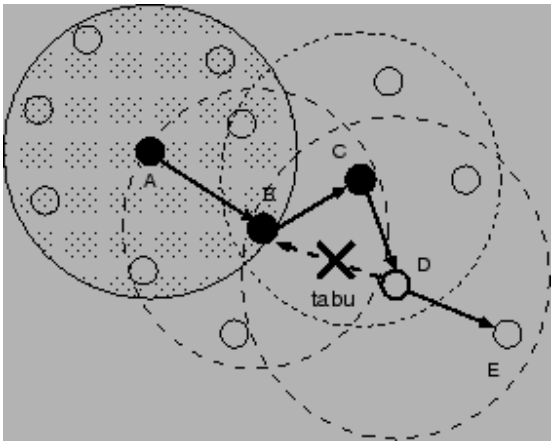

Figura 6: Exemplo de um Movimento Tabu

Duas regras são normalmente utilizadas de forma a interromper o procedimento. A primeira é quando é atingido o número máximo de iterações sem melhora no valor da melhor solução e a segunda, quando o valor da melhor solução chega a um limite inferior conhecido (ou próximo dele).

Os principais parâmetros de controle do método de Busca Tabu são a cardinalidade |T| da lista tabu, a função de aspiração A, a cardinalidade do conjunto V de soluções vizinhas testadas em cada iteração e BTmax, o número máximo de iterações sem melhora no valor da melhor solução.

Abaixo o procedimento Busca Tabu:

Procedimento BT (f(.), N(.),A(.),|V| , $S_{min,}$ |T| , BTmax, s)

1 s*←s; //Melhor solução obtida até então

2 IterT ←0;//número de iterações na temperatura T

3 MelhorIter ←0;//Iteração mais recente que forneceu S*

4 T = 0;//Lista Tabu

5 Inicialize a Função de Apiração A

6 enquanto (f(S) > f_{min} e Iter – MelhorIter < BTmax) faça

7 IterT = IterT +1;

8 Seja s' ←s $\oplus$ m o melhor elemento de V $\subset$ N(S) tal que o movimento m não seja Tabu m$\notin$ T ou S' atenda a condição de aspiração (f(s') < A(f(s)));

9 T ← T – {Move mais antigo} + {mov. Que gerou S'};

10 Atualize a função de aspiração A;

11 s ← s';

12 se f(s') < f(s*) então

13 s* ← s';

14 MelhorIter ← Iter

15 fim-se

16 fim-enquanto

17 s ← s*;

18 Retorne S

19 fim BT

2.4.3. GRASP

O método GRASP (Greedy Randomized Adaptive Search Procedure) utiliza-se de uma heurística gulosa para compor a solução inicial de cada iteração, com o intuito de conduzir o algoritmo a soluções próximas do ótimo local, acelerando a busca local (PRAIS; RIBEIRO, 1999; FEO; RESENDE, 1995). O princípio deste método é a combinação de método construtivo e busca local, sendo um procedimento iterativo com completa independência de iterações.

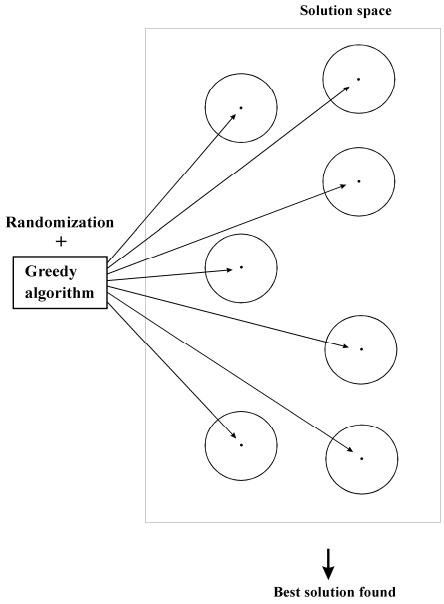

Figura 7: Exemplo de Movimentação no GRASP

Algoritmo GRASP

Passo 1. Fixar c(S*) ←∞

Passo 2. **Para** Nmax de iterações **faça**

Construir uma solução S usando um algoritmo guloso aleatório

Aplicar um procedimento de busca local começando em S e finalizando em S'

Se C(S') melhorar a melhor solução corrente S^*, ou seja, se C(s') < C(s*) então

Fixar S* ←S'

Passo 3. Imprimir a melhor solução S*

Capítulo 3

Computação Evolucionária

3.1 Uma Visão Geral

Algumas modelagens de problemas matemáticos podem levar a funções bastante complexas, multidimensionais, não lineares e descontínuas. A maximização ou minimização dessas funções através de métodos matemáticos convencionais, a fim de se otimizar a solução do problema, pode ser muito difícil de se resolver. Logo as técnicas de algoritmos evolucionários – Algoritmos genéticos, colônia de formigas, etc. – podem ser empregadas para resolver esse tipo de problema.

As técnicas de computação evolucionária são motivadas pela evolução da natureza, onde uma população de indivíduos, os quais pertencem à solução do problema, são manipulados de acordo com as regras de sobrevivência gerando uma melhor solução que é disseminada através das gerações.

3.2 Algoritmos Evolucionários

Os Algoritmos Evolucionários (AE) são metodologias da área da computação evolucionária, não são algoritmos computacionais em seu significado usual, mas formam uma classe de métodos regidos por princípios similares, que tentam abstrair e imitar alguns dos mecanismos evolutivos à resolução de problemas que requerem adaptação, busca e otimização. Os Algoritmos buscam uma solução ótima em um determinado domínio a partir de um conjunto de possíveis soluções que são testadas e comparadas entre si. Em função dessa comparação, essas soluções são submetidas a um processo de evolução e seleção, onde as melhores soluções são guardadas. Este processo evolui até que a melhor das soluções

tentativas atinja um determinado patamar de desempenho ou simplesmente até que um determinado número de evoluções tenha ocorrido.

O algoritmo básico da computação evolucionária pode ser visto na Figura 8.

Algoritmo Evolucionário Conceitual
Inicialização da população
Avaliação dos indivíduos na população
Repetição (evolução)
Seleção dos indivíduos da população
Alteração desses indivíduos para criar uma nova população
Avaliação da nova população
Até que um critério de parada seja alcançado
Fim

Figura 8: Algoritmo Básico da CE

Os processos naturais, mais comuns, aos quais os algoritmos evolucionários se baseiam são os processos genéticos da evolução das espécies e os processos que orientam a movimentação de enxames (*swarm*) de insetos.

As técnicas de computação evolucionária operam sobre uma população de candidatos em paralelo. Assim, elas podem fazer a busca em diferentes áreas do espaço de solução, alocando um número de membros apropriado para a busca em várias regiões.

A seguir serão descritos dois exemplos clássicos da computação evolucionária os algoritmos genéticos e colônia de formigas.

3.2.1 ALGORITMOS GENÉTICOS

Os algoritmos genéticos (AG) (TOMASSINI, 1998) como uma forma de resolver problemas de busca adaptativa no qual o conhecimento para controlar a procura é obtido dinamicamente. O ponto de vista de que os seres vivos evoluem de acordo com o princípio da seleção natural, defendido por Charles Darwin no século XIX, é hoje consensualmente aceita. A teoria da seleção natural foi complementada com os estudos sobre genética iniciados por Mendel. Em termos gerais este paradigma estipula que:

- Cada indivíduo transmite aos seus descendentes os seus traços característicos (genéticos);
- Numa população existem indivíduos com traços (características) diferentes;
- Os indivíduos com melhores características (mais adaptados) tendem a reproduzir-se mais, pelo que uma população caminha no sentido da existência em maior número dos elementos mais adaptados ("mais fortes");
- Ao longo do tempo o acúmulo de pequenas variações pode originar o aparecimento de novas espécies com melhor capacidade de sobrevivência;
- Processos de troca e aparecimento de material genético (novo) podem concorrer para uma melhor adaptabilidade.

Existem três mecanismos de base para explicar a dinâmica das populações:

- **reprodução adaptativa**: as espécies reproduzem-se aumentando o número dos mais aptos ("fitness");
- **recombinação** ("crossover"): novas combinações de genes existentes;
- **mutação** ("mutation"): aparecimento de novos genes.

Em termos gerais a passagem de uma geração a seguinte faz-se de acordo com o princípio da reprodução adaptativa, perturbado pelos mecanismos de recombinação e mutação.

Os algoritmos genéticos são uma versão computacional, simplificada, do que se passa na natureza, sendo necessária a identificação de um conjunto de elementos e pela definição de um conjunto de valores, a saber:

- **Geração** = uma população num dado instante do tempo;
- **População** = conjunto de indivíduos;
- **Indivíduo** = definido pelos seus cromossomos (tipicamente um);
- **Cromossomo** = conjunto de genes;
- **Gene** = codificação de uma parte do cromossomo podendo tomar diferentes valores;
- **Função de adaptabilidade**: para medir a qualidade do indivíduo;
- **Recombinação**: qual a probabilidade de haver troca de material genético entre dois indivíduos;
- **Mutação**: qual a probabilidade de aparecer material genético novo? Geralmente um valor baixo.

Os Algoritmos Genéticos (AG's) diferem-se dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos:

- Trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros;
- Trabalham com uma população e não com um único ponto;
- Utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar;
- Utilizam regras de transição probabilísticas e não determinísticas.

O funcionamento do algoritmo genético ocorre basicamente do seguinte modo:
Inicialmente, é gerada uma população formada por um conjunto aleatório de indivíduos que podem ser vistos como possíveis soluções do problema. Durante o processo evolutivo, esta população é avaliada: para cada indivíduo é dada uma nota, refletindo sua habilidade de adaptação a um determinado ambiente. Uma porcentagem dos mais adaptados é mantida, enquanto os outros são descartados (darwinismo). Os membros mantidos pela seleção podem sofrer modificações em

suas características fundamentais através de mutações e cruzamento (crossover) ou recombinação genética gerando descendentes para a próxima geração. Este processo, chamado de reprodução, é repetido até que uma solução satisfatória seja encontrada.

Embora possam parecer simplistas do ponto de vista biológico, estes algoritmos são suficientemente complexos para fornecer mecanismos de busca adaptativos poderosos e robustos.

```
Algoritmo Genético
- Inicio
      t = 0;
      inicia_população (P, t)
      avaliação (P, t);
      Repita até (t = d)
            Início
                  t = t +1;
                  seleção_dos_pais (P,t);
                  recombinação (P, t);
                  mutação (P, t);
                  avaliação (P, t);
                  sobrevivem (P, t)
            Fim
- Fim
```

onde:
t - tempo atual;
d - tempo determinado para finalizar o algoritmo;
P - população

Figura 9: Algoritmo Básico do AG

3.2.2 COLÔNIA DE FORMIGAS

O princípio da colônia de formigas é a simulação do comportamento de um conjunto de agentes que cooperam para resolver um problema de otimização por meio de comunicações.

Formigas parecem achar o caminho delas (do ninho para uma fonte de comida e atrás, ou ao redor de um obstáculo) com uma relativa facilidade, como mostra a figura 10. Estudos descobriram que em muitos casos esta capacidade é o resultado da comunicação química entre as formigas e um fenômeno emergente causado pela presença de muitas formigas. Cada formiga age da seguinte forma:

- Em todo passo as formigas escolhem, usando uma regra probabilística, uma cidade ainda não visitada;
- A probabilidade de escolher o nó (i, j) é diretamente proporcional a quantidade Tij de t de *pherormone* no nó (i, j) e para o inverso do nó de tamanho Cij;
- Cada formiga lembra-se de cada nó visitado;
- Depois de uma excursão ter sido completada, as formigas põem um rastro positivo ΔTij em cada nó (i, j) da excursão (ΔTij = 0 para os nós que não estão na excursão).

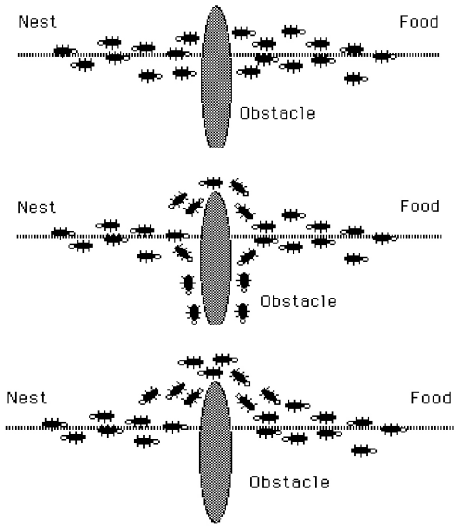

Figura 10: Exemplo do Caminho Obtido pelas Formigas

Desde que os valores iniciais de T_{ij} são constantes para todo nó, inicialmente as formigas são livres para mover-se aleatoriamente. Se nós consideramos M formigas movendo-se em um grafo simultaneamente, T_{ij} é o caminho pelas quais as formigas se comunicam. As formigas escolhem com maior probabilidade, nós com maiores quantidades de *pherormone*. Então, as formigas deixam rastros nos nós fazendo-os mais desejáveis para outras formigas. O algoritmo da colônia de formiga pode ser declarado como a superposição simultânea de M procedimentos de formigas.

Algoritmo da colônia de formigas

Passo 1. **Seja** Q e τ_0 constantes, e **fazer** $c(S^*) \leftarrow \infty$

Passo 2. **Fazer** $\Delta\tau_{ij} \leftarrow 0$ e $\tau_{ij} \leftarrow \tau_0$ **para todo** nó (i,j)

Passo 3. **Para** cada formiga k = 1,2,..., M **Faça**

Selecione a cidade inicial da qual a formiga k parte

Obtenha uma excursão T^k para cada formiga k de acordo com o procedimento seguido por cada formiga individual

Seja L^k ser a duração de excursão T^k

Se $L^k < c(S^*)$, **então**

$S^* \leftarrow T^k$

Compute a quantidade do rastro deixado pela formiga k **como**:

$\Delta\tau_{ij}(k) = Q / L^k$, se o nó (i,j) pertence a excursão T^k,

caso contrário $\Delta\tau_{ij}(k) = 0$

Fazer $\Delta\tau_{ij} \leftarrow \Delta\tau_{ij} + \Delta\tau_{ij}(k)$

Passo 4. **Fazer** $\tau_{ij} \leftarrow \beta\tau_{ij} + \Delta\tau_{ij}$

Passo 5. Se a melhor excursão S* não mudou para o último Nmax global de iterações, **então**

pare com S* como a melhor solução encontrada,

caso contrário

volte ao passo 2.

Figura 11: Algoritmo da AC

Capítulo 4

Otimização por Enxame de Partículas

4.1 Introdução

Uma população de indivíduos, os quais representam possíveis soluções para o problema, são manipulados de acordo com as regras de sobrevivência que melhor adaptam-se ao meio a partir de modificações proporcionadas por operadores genéticos. Tais como: mutação, cruzamento e reprodução, fazendo com que a melhor solução evolua através das gerações. Baseando-se nas técnicas evolucionárias, Eberhart e Kennedy desenvolveram um algoritmo diferente, através da simulação do comportamento social (KENNEDY;EBERHART, 2001). Este algoritmo foi chamado de Particle Swarm Optimization (PSO) ou Otimização por Enxame de Partículas, pois foi inspirado no comportamento localmente aleatório, mas globalmente centralizado, de um bando de pássaros durante seu vôo. No PSO, assim como em outros algoritmos, existe uma população de indivíduos, chamados partículas, mas, ao invés de utilizar operadores genéticos, as partículas “evoluem” através da cooperação e competição entre elas mesmas por diversas “gerações” (iterações).

As partículas se beneficiam de sua própria experiência e da experiência de outros membros do enxame durante a busca por local melhor. A Figura 12 mostra uma analogia entre a busca dos pássaros em busca de um puleiro para dormir, representado pelo círculo vermelho. E a busca por um ótimo local, onde temos o espaço de busca, os ótimos locais (áreas verdes) e o melhor global círculo vermelho.

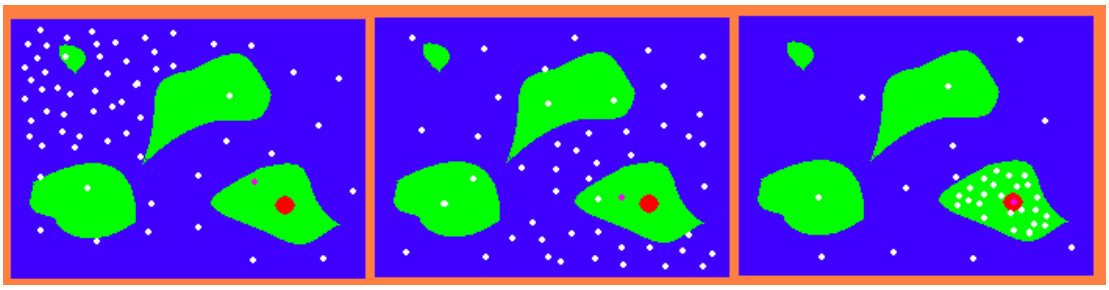

Figura 12: Analogia entre Busca de um Puleiro e Busca de um Ótimo Global

4.2 Estrutura do Algoritmo

Em sua versão original o *PSO* é muito similar a algumas técnicas da computação evolucionária, como os algoritmos genéticos (*GA*), onde o sistema é inicializado com uma população de soluções randômicas. Entretanto, o PSO difere de outras técnicas com populações bem estabelecidas, pois nenhum operador inspirado pelos procedimentos de DNA é aplicado na população para extrair uma geração nova de candidatos a soluções. Em vez dos procedimentos de DNA, são utilizadas *partículas (*ver Figura 13*)*, movendo-se em um espaço de busca n-dimensional, sendo cada uma delas uma solução potencial para o problema. Cada partícula possui também uma velocidade randômica para que possa percorrer o espaço problema. A i-ésima partícula é representada tendo o D representando a dimensão:

$$X_i = (X_{i1}, X_{i2},\ldots, X_{iD}).$$

A melhor posição encontrada, aquela com o melhor valor de aptidão encontrado pela i-ésima partícula, é registrada e representada como:

$$P_i = (P_{i1}, P_{i2},\ldots, P_{iD}).$$

O índice da melhor partícula entre todas as partículas na população é representado pelo símbolo *g*. A taxa da mudança de posição, que é chamada de velocidade, para partícula i é representada como:

$$V_i = (V_{i1}, V_{i2},\ldots, V_{iD}).$$

A movimentação de cada partícula é baseada em três parâmetros: *fator de sociabilidade* que determina a atração das partículas para a melhor posição descoberta por qualquer elemento do enxame; *fator de individualidade* que determina a atração da partícula com sua melhor posição e a *velocidade* que determina deslocamento da partícula, velocidade é restringida para que as partículas não saiam do seu espaço de busca.

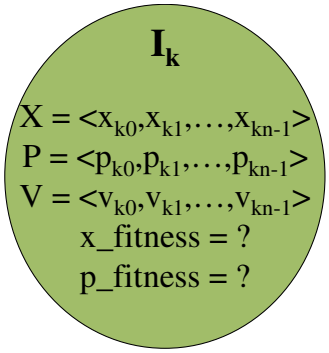

Figura 13: Representação de uma Partícula

As partículas são manipuladas de acordo com as seguintes equações:

(1) $V_{id} = W^* V_{id} + C_1 * rand\ () * (P_{id} - X_{id}) + C_2 * Rand\ () * (P_{gd} - X_{id})$

(2) $X_{id} = X_{id} + V_{id}$

Onde:

C_1 e C_2 - são duas constantes positivas que correspondem as componentes cognitivas e sociais (individualidade e sociabilidade);

rand () e Rand () - são duas funções aleatórias no intervalo [0,1];

W - é o peso de inércia.

A Equação (1) é usada para calcular a nova velocidade da partícula de acordo com sua velocidade anterior e as distâncias entre sua posição atual, sua melhor posição e a melhor posição do grupo. A partir daí, a partícula “voa” para uma nova posição de acordo com equação (2). O desempenho de cada partícula é medido de acordo com uma função de aptidão pré-definida que é relacionada ao problema a ser resolvido. O peso de inércia W é empregado para controlar o impacto da

velocidade anterior na velocidade atual, influenciando assim as habilidades de exploração global e local das partículas. Um peso de inércia maior facilita a procura por novas áreas dentro do espaço um peso de inércia menor tende a facilitar exploração local para refinar a área de procura atual. Com as execuções observou-se que a seleção satisfatória do peso de inércia W pode prover um equilíbrio entre habilidades de exploração global e local, podendo dessa forma requerer menos repetições, em média, para encontrar o valor ótimo.

Cada partícula mantém o rastro de suas coordenadas no espaço problema, que estão associadas à melhor solução (*fitness*) que ela tenha encontrado até então. O valor do *fitness* também é armazenado. Esse valor é chamado *pbest*. Outro valor armazenado é o melhor sobre todos os valores, e sua posição, obtido por qualquer partícula na população. Essa posição é chamada de *gbest*.

O conceito do *PSO* consiste de, a cada passo, ocorra a mudança de velocidade (aceleração) e as partículas alcancem suas posições de *pbest* e *gbest.* O processo original para implementação da versão global do PSO é realizado de acordo com a Figura 14:

P o tamanho da população do PSO.
PSO [i] a posição da i-ésima partícula da população do PSO, que representa uma solução candidata para o problema.
fitness[i] o custo da função da i-ésima partícula.
V[i] a velocidade da partícula.
G_{best} um índice para a melhor posição global.
$P_{best}[i]$ a posição de melhor localização da i-ésima partícula.
$P_{best\_fitness}[i]$ o melhor fitness local visitado pela i-ésima partícula.
Passo 1 (Inicialização): Para cada partícula i na população:
Passo 1.1: Inicialize o *PSO[i]* randomicamente.
Passo 1.2: Inicialize *V[i]* randomicamente.
Passo 1.3: Avalie o *fitness[i]*.
Passo 1.4: Inicialize $G_{best.}$
Passo 1.5: Inicialize *Pbest[i]* com uma cópia de $PSO[i] \ \forall \ i \leq P$.
Passo 2: Repita até que um critério de parada seja satisfeito
Passo 2.1: Encontre o G_{best} tal que $fitness[G_{best}]$ $fitness[i]$ $i \leq P$.
Passo 2.2: Para cada partícula *i*:
$P_{best}[i] = PSO[i]$ *se* $fitness[i] > P_{best\_fitness}[i] \ \forall \ i \leq P$.
Passo 2.3: Para cada particular *i*:
Atualize *V[i]* e *PSO[i]* de acordo com as equações 1 e 2.
Passo 2.4: Avalie o $fitness[i] \ \forall \ i \leq P$

Figura 14: Algoritmo Clássico do PSO

O funcionamento do algoritmo resume-se em:

1. Inicialização de uma população de partículas com posições e velocidades aleatórias em um espaço de busca d-dimensional.
2. Para cada partícula, avalia-se a função de fitness.
3. Compara-se o fitness e o pbest da partícula. Se o valor atual for melhor que o pbest então, o valor do pbest é substituído pelo valor do fitness encontrado, e a posição do pbest pela posição atual no espaço d-dimensional.
4. O valor do fitness é comparado com o melhor valor encontrado em toda a população. Se o valor atual for melhor que o gbest então, o gbest é modificado para o índice e valor da partícula atual.
5. Modifica-se então a velocidade e posição da partícula de acordo com as equações (1) e (2) respectivamente.
6. Executa-se um laço para o passo 2 até que um critério seja encontrado: normalmente, um fitness bom o suficiente ou um número máximo de iterações seja executado.

4.3 PSO Discreto

As modificações propostas para o *PSO discreto* podem ser ditas de certa forma, elegantes, pois elas preservaram toda a estrutura do algoritmo *PSO original* e, além disso, inseriu o *PSO discreto* em uma nova classe de problemas. As mudanças sugeridas na literatura, foram: Os vetores de posição atual e melhor posição foram substituídos por vetores com valores discretos, e o vetor velocidade, por uma lista de trocas, onde essas trocas serão aplicadas ao vetor posição para a obtenção de uma nova posição.

4.3.1 População Inicial

As partículas no *PSO discreto* são representadas através de arestas/nós, sendo cada um destes uma dimensão, representadas em um vetor posição.

Figura 15: Representação de uma Partícula no PSO Discreto

4.3.2 Operadores Discretos

4.3.2.1 Velocidade

No espaço discreto, a velocidade não é mais somente um escalar. Ela deve ser um operador que, quando aplicado a uma posição, resulte em outra posição. Sendo assim, a velocidade é definida como uma lista de N transposições: Onde velocidade = $\{(i_1,j_1),(i_2,j_2), \ldots (i_N,j_N)\}$, que significa a troca dos vértices de índices i_1 e j_1, i_2 e j_2, etc. até a troca dos vértices i_N e j_N.

A velocidade pode ser dita equivalente, ou seja, velocidade $v_1 = v_2$, se nós aplicarmos v_1 ou v_2 em uma posição e obtivermos o mesmo resultado. Uma velocidade nula é uma lista vazia, ∅. $|v|$ denota o tamanho da velocidade *v*, que é igual ao número de transposições em *v*.

4.3.2.2 Movimentação (Posição + Velocidade)

Seja *P* uma posição e *V* uma velocidade, logo, *P'* será uma nova posição onde *P' = P + v*, onde devem ser aplicadas todas as transposições pertencentes a *V*.

Por exemplo, seja *P=[2,3,4,5,1], V = {(1,2), (2,4)}*, a operação é realizada da seguinte forma: Aplica-se a transposição *(1,2)*, ou seja, troca-se o conteúdo da 1ª célula e da 2ª célula em *P*, onde será obtido P_1*=[3,2,4,5,1]*. Em seguida, aplica-se

a transposição *(2,4),* ficando P_2*=[3,5,4,2,1]*. Como a última transposição foi alcançada, uma nova posição *P'* é obtida: *P'=*P_2*=[3,5,4,2,1]*.

4.3.2.3 Obtenção da Velocidade (Posição – Posição)

Sejam *P1* e *P2* duas posições. A diferença *P1 - P2* é definida como sendo a velocidade *V*, onde o sinal de " - " possui um novo significado. Através dessa diferença obteremos a lista de transposições, resultando na seguinte equação:

$P_2 = P_1 + v$.

Como podemos verificar no exemplo logo abaixo:

Exemplo: Dadas duas posições *P1* e *P2*:

P_1*= [1 2 3 4 5]* P_2*= [2 3 1 5 4]*

P_1*[1] =* P_2*[3] = 1*, logo a primeira transposição será *(1,3)*, então teremos um P_2*'=* P_2 *+ S(1,3)*. Portanto, P_2*' = [1 3 2 5 4]*

P_1*[2] =* P_2*'[3] = 2*, logo a segunda transposição será *(2,3)*, então teremos um P_2*''=* P_2*' + S(2,3)*. Logo, P_2*'' = [1 2 3 5 4]*

Seguindo na seqüência, o terceiro operador será *(4,5)*, logo será obtida a lista de transposição *(v)*, que será igual a:

$P_1 - P_2$ *= {(S(1,3), S(2,3), S(4,5))}.*

4.4.2.4 Adição entre Velocidades (Velocidade + Velocidade)

A adição entre duas velocidades consiste na concatenação de listas de transposições. Resultando uma nova lista.

4.4.2.5 Multiplicação (Coeficiente * Velocidade)

Seja *C* um coeficiente e *V* uma velocidade. São possíveis quatro casos:

(i) *C* = 0, teremos *C*V* = ∅;

(ii) $C \in [0,1]$, truncamos *V* por *(1-C) *|V|*;

(iii) $C > 1$, aplicamos v, c vezes para a parte inteira e aplicamos (ii) para a parte fracionária;

(iv) *C<0, não é definido para esta implementação.*

Capítulo 5

Estratégias Utilizadas

Para realização dos testes, o PSO foi aplicado a 16 problemas da TSPLIB variando entre 35 a 3038 cidades, tanto problemas simétricos como assimétricos. Nas execuções seqüenciais os algoritmos foram implementados em Visual Basic e já implementação dos algoritmos paralelos fora utilizada a linguagem C++, utilizando a biblioteca MPI (Message-Passing Interface), que consiste em padrão de interface para troca de mensagens em máquinas paralelas com memória distribuída.

Primeiramente, com a execução do algoritmo em apenas uma máquina, fora executado o algoritmo com o PSO clássico (versão discreta, sem nenhuma melhoria), este não apresentou resultados satisfatórios, sendo necessário o emprego de estratégias mais sofisticadas. Segue abaixo algumas das estratégias, mais relevantes, utilizadas e os resultados da implementação das mesmas:

5.1 – Introdução de Buscas Locais

Foi verificado, durante as execuções, que as partículas estavam ficando “presas“ em ótimos locais, prematuramente, não encontrando mais nenhum valor melhor. As partículas seguiam para a posição da melhor partícula do enxame e não saiam dessa posição. Então foi aplicada uma busca local apenas para as partículas que melhorassem de fitness, verificando uma pequena melhora nos resultados, mas ficando ainda aquém do resultado ótimo, resultado este previamente conhecido. Em seguida foi aplicada uma busca local para todas as partículas onde foi verificada uma melhor resposta cerca de 20% ainda distante do ótimo, para as instâncias pequenas (abaixo de 70 cidades), porém a execução ficava bastante lenta, já que a busca local utilizada era a troca de posições entre os nós, como

mostra a Figura 16, partindo daí a necessidade da alteração da busca local utilizada.

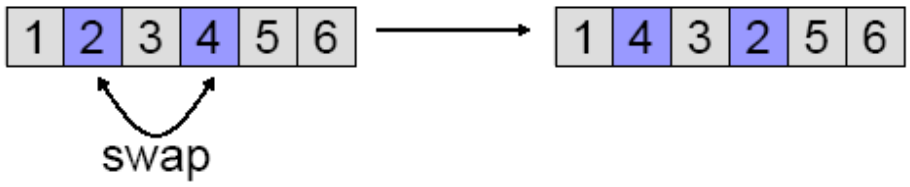

Figura 16: Busca Local Swap

5.2– Aplicação dos Bairros

Apesar da melhora obtida com a busca local, as partículas ainda apresentavam o problema de ficarem presas a ótimos locais. Então foi aplicado o conceito de bairros nesses enxames, separação das partículas em grupos, estes denominados de bairros, que eram interligados através de algumas partículas que fossem pertencentes a mais de um bairro fazendo com que as informações fluíssem entre os mesmos, conforme a figura 17. O problema das partículas ficarem presas a ótimos locais foi amenizado devido a uma maior variabilidade de resultados encontrados pelas partículas e os resultados ficaram um pouco melhores, porém os resultados ainda ficavam ruins e apenas retardavam a queda em ótimos locais.

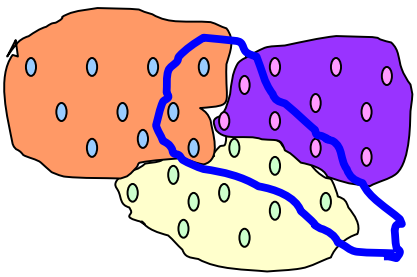

Figura 17: Bairros Interligados

5.3– Desconectando os Bairros

Em relação à estrutura, fora realizada a desconexão desses bairros do seguinte modo: as partículas foram separadas em bairros desconexos. Cada bairro desses é considerado um enxame individual. Após certo número de iterações, é selecionado as melhores e as piores partículas de cada bairro para então a realização de algumas iterações com essas partículas em um novo bairro. Logo depois as mesmas são colocadas de volta nos seus bairros a fim de fazer com que todas as partículas saibam onde está o melhor valor de todo o enxame, repetindo esta operação até o critério de parada do algoritmo. Com esta modificação foram encontrados os melhores resultados, até então, chegando a atingir os valores ótimos das instâncias aplicadas ao estudo.

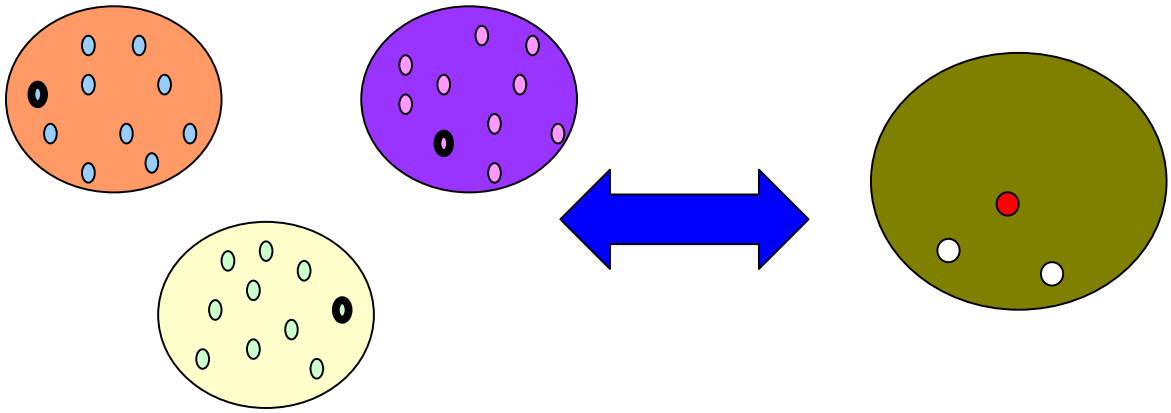

Figura 18: Bairros Separados

5.4– Nova Obtenção da Velocidade

Para melhorar a eficiência do algoritmo foi realizada uma modificação na forma da obtenção da velocidade e aplicação da mesma. Ao invés da velocidade ser obtida através da subtração de duas posições para a obtenção de uma lista de velocidade (lista de transposições), a velocidade passa a ser obtida através das arestas que se encontram nas melhores posições da partícula (melhor vizinho M.V) e a melhor posição encontrada até o presente momento do algoritmo (M.P).

A partir das arestas que se encontram nessas rotas e não se encontra na rota atual (R.A) da partícula é que será formada a nova posição da partícula, como pode ser visto na Figura 19. Com isto o algoritmo ganhou bastante em velocidade na convergência para os melhores valores encontrados pelos enxames, chegando próximo ao valor ótimo em alguns casos.

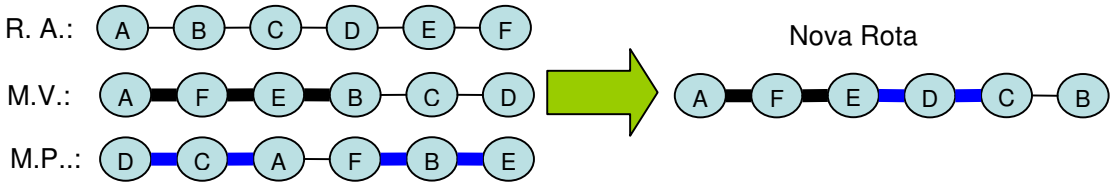

Figura 19: Geração de uma Nova Rota

5.5 – 2-Opt + VND

Finalmente, foi aplicado a busca local 2-Opt (RIBEIRO, 1998) nas partículas que melhorassem no decorrer das iterações. Isto porque, a mesma já é bem conhecida na literatura por obter bons resultados aplicada ao problema do caixeiro viajante. Uma busca do tipo VND (RIBEIRO, 1998) nas iterações com as melhores partículas de cada bairro, onde foram aplicadas três diferentes tipos de busca local, entre essas a 2-Opt, a busca local Insert e uma busca local, chamada de merge, que escolhia uma sub-rota aleatória de sua melhor vizinha e aplicava na rota atual na tentativa de melhorar o fitness da partícula.

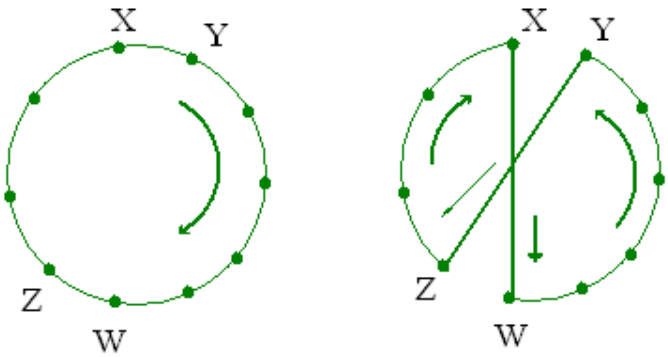

Figura 20: Busca Local 2-Opt

Após a aplicação destas estratégias no algoritmo seqüencial do PSO as estratégias implementadas serviram de base para a aplicação do algoritmo do PSO, em um ambiente paralelo. Pois o algoritmo chegou a um ponto de "maturidade", onde já se conhecia o comportamento do mesmo com as estratégias aplicadas, além da busca contínua por resultados melhores. O objetivo seria a melhora dos resultados obtidos, devido principalmente a variabilidade das possíveis soluções proporcionada pelo paralelismo, além da performance das repostas, possível graças as máquinas e redes com boas configurações, utilizando duas máquinas quad-cores 2.4Ghz com 2Ghz de memória. Os algoritmos paralelos têm sido usados para resolver problemas de grande porte, o que se traduz diretamente na tentativa de reduzir os altos custos computacionais envolvidos. Para isto, a idéia básica é dividir a carga de processamento de uma determinada aplicação de forma a permitir que vários processadores possam auxiliar em sua execução.

A priori foram desenvolvidas duas estratégias básicas para aplicação do PSO paralelo, nos dois casos fora utilizada a estrutura e rotina do PSO "original", mas com a aplicação da busca local insert e 3-Opt, para as partículas que melhorassem de posição.

A duas primeiras estratégias utilizadas no algoritmo PSO-Paralelo são:

5.6– Mestre x Escravo com 3-Opt

Na primeira estratégia foi utilizado um esquema de Mestre X Escravos, conforme Figura 21. A melhor partícula de cada escravo é enviada ao mestre após certo número de iterações (foi utilizado 200 iterações). O mestre por sua vez substitui suas piores partículas pelas partículas enviadas pelos escravos. Foi percebida uma melhoria bastante significativa tanto em tempo de resposta, caindo da casa

dos minutos para segundos, quanto em resultados se comparado a execução do PSO de forma seqüencial, os dados serão exibidos no próximo capítulo.

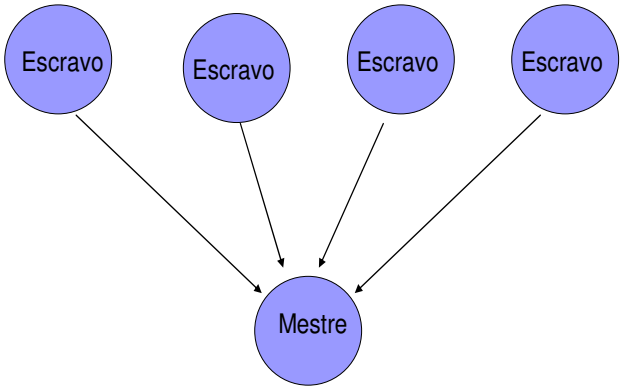

Figura 21: Esquema Mestre X Escravos

5.7– Broadcast

Na segunda estratégia visando uma maior variabilidade entre os enxames de cada máquina em execução, todos os enxames trocam informações entre si. Eliminando as “n” piores partículas de cada enxame e as substituindo pelas melhores partículas de cada enxame em execução, sendo “n” a quantidade de ilhas, conforme Figura 22 com n = 5. Como o esperado os resultados melhoraram em relação a estratégia anterior, devido a diversidade de posição das partículas com as trocas mútuas entre os enxames.

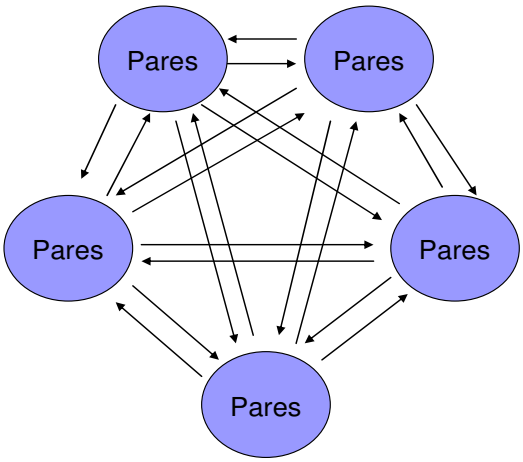

Figura 22: Transmissão mútua entre os Enxames

Após vários testes com as duas estratégias descritas anteriormente, como aplicação de diversas buscas locais, testes com diferentes modos de comunicação entre mestre e escravos, sempre tentando encontrar uma melhor forma entre a relação Resultado X Tempo, foram selecionadas mais duas estratégias para fechar este conjunto de idéias.

5.8– Mestre x Escravo com Path-Relinking

Na terceira estratégia fora mantida a estrutura Mestre X Escravos. As melhores partículas dos escravos são enviadas ao mestre, substituindo as “n” piores partículas do mestre e estas “n” piores partículas do mestre substituem as “n” piores partículas dos escravos, sendo “n” a quantidade de escravos. Mas antes de das partículas serem substituídas é aplicado o algoritmo Path-Relinking (SOUZA, 2006) entre as duas partículas tendo como rota a ser seguida sempre a partícula que chega no enxame. A idéia deste algoritmo é deixar o mestre com uma população de possíveis soluções com bons resultados e aumentar a variabilidade nos escravos.

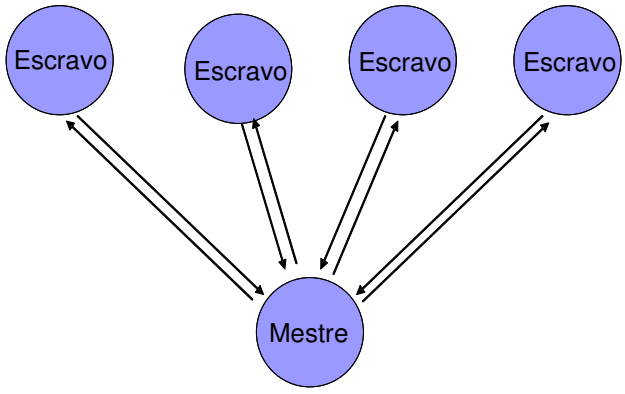

Figura 23: Estratégia Mestre X Escravos, Path-Relinking

5.9– Mestre x Escravo Intercalando Busca Local

A última estratégia foi baseada em uma estratégia apresentada em (FILHO,2008) utilizando a estrutura Mestre X Escravos. Semelhante a estratégia anterior as melhores partículas dos escravos são enviadas ao mestre, chegando ao mestre este analisa a partícula que acabara de chegar verificando se é melhor que a sua melhor partícula. E em caso positivo, atualiza sua melhor partícula, em seguida, envia sua partícula substituída de volta para o escravo, onde este assim que receber realizará o Path-Relinking com sua melhor partícula para enviar o resultado novamente ao mestre que estará fazendo a comparação com o escravo n+1, seguindo o ciclo até o último escravo.

Capítulo 6

Resultados

Nesta seção encontram-se as tabelas com as médias e os melhores resultados, obtidos nos testes. Quanto a parte seqüencial, foram realizadas 10 execuções para cada uma das estratégias, com limite máximo de 45 minutos, com 200 partículas. As instâncias que se encontram com “XX” ficaram com o tempo inviável de execução, ultrapassando o limite tempo estabelecido, mas continuaram a ser estudadas com o objetivo de serem obtidos resultados melhores para as mesmas e quanto a execução do PSO-Paralelo as mesmas foram executadas com apenas 40 partículas em cada enxame, sendo transmitidas as partículas para o mestre (ou entre si) a cada 200 iterações, limitadas a 20 minutos.

Podemos observar uma melhoria tanto na obtenção dos resultados quanto em relação ao tempo de execução para os algoritmos com a introdução de cada nova estratégia, por exemplo da melhoria do fitness alcançado podemos citar a instância ftv170, que saiu de um gap de 531%, acima do ótimo conhecido para 8.5%, em média, saindo de uma execução que durava cerca de 15 minutos para poucos segundos, cerca de no máximo 50 segundos. Já em relação ao tempo de execução temos de forma bem clara a execução das instâncias rbg323 e rbg403, que nas execuções seqüenciais, não conseguiram atingir um bom resultado no tempo limite de 45 minutos e nas implementações em paralelo conseguindo ficar próximo do resultado ótimo em média e até mesmo atingi-lo. Logo adiante seguem as tabelas com as médias e os melhores resultados obtidas por cada estratégia.

Tabela 1: Médias dos Resultados Seqüenciais

Instância	PSO Clássico	gap(%)	Estratégia 1 (Bairros Ligados e Busca Local)	gap(%)	Estratégia 2 (Bairros Separados e Busca Local)	gap(%)	Estratégia 3 Mud. Velocidade	gap(%)	Insert	gap(%)	2-Opt e VND	gap(%)	Ótimo
ftv35	3253,5	**120,88**	1710,3	**16,11**	1657,4	**12,52**	1585,38	**7,63**	1536	**4,28**	1509,1	**2,45**	**1473**
p43	6429,5	**14,4**	5667,3	**0,84**	5641,6	**0,38**	5636,16	**0,29**	5634	**0,25**	5621,5	**0,03**	**5620**
brasil58	90636,2	**287,42**	30291,3	**29,48**	29352	**25,46**	27649,66	**18,19**	26472	**13,15**	25413	**8,63**	**23395**
ftv70	7813	**300,67**	2822,1	**44,72**	2874,7	**47,42**	2535,04	**30**	2259	**15,87**	2184,5	**12,03**	**1950**
kro124	112087	**209,38**	52940,5	**46,12**	50716,4	**39,98**	49197,78	**35,79**	44717	**23,43**	40736	**12,44**	**36230**
ftv170	17403,8	**531,72**	6623	**140,4**	6355,4	**130,69**	5220	**89,47**	4584	**66,4**	4040	**46,64**	**2755**
rbg323	5715,5	**331,03**	1581,1	**19,24**	1564,2	**17,96**	1548,86	**16,81**	1482	**11,78**	XX	**XX**	**1326**
rbg403	7146,6	**189,92**	2945,7	**19,5**	2762	**12,05**	2616,42	**6,14**	2518	**2,15**	XX	**XX**	**2465**

Tabela 2: Melhores Resultados Seqüenciais

Instância	PSO Clássico	gap(%)	Estratégia 1 (Bairros Ligados e Busca Local)	gap(%)	Estratégia 2 (Bairros Separados e Busca Local)	gap(%)	Estratégia 3 Mud. Velocidade	gap(%)	Insert	gap(%)	2-Opt e VND	gap(%)	Ótimo
ftv35	3099	**110,39**	1644	**11,60**	1542	**4,68**	1490	**1,15**	1487	**0,95**	1473	**0,00**	**1473**
p43	6118	**8,87**	5655	0,63	5624	0,07	5621	**0,01**	5623	**0,05**	5620	**0,00**	**5620**
brasil58	84872	**262,78**	30270	**29,38**	27551	**17,76**	25732	**9,98**	25395	**8,55**	25395	**8,55**	**23395**
ftv70	7431	**281,07**	2797	**43,46**	2730	**40,00**	2259	**15,84**	2030	**4,10**	2021	**3,64**	**1950**
kro124	105832	**192,11**	50034	**38,10**	48385	**33,54**	47705	**31,67**	41137	**13,54**	38564	**6,44**	**36230**
ftv170	16580	**501,82**	6267	**127,48**	5995	**117,60**	4795	**74,00**	3985	**44,65**	3480	**26,32**	**2755**
rbg323	5641	**325,42**	1549	**16,81**	1517	**14,40**	1477	**11,38**	1431	**7,92**	XX	**XX**	**1326**
rbg403	7031	**185,23**	2733	**10,87**	2692	**9,21**	2544	**3,20**	2481	**0,65**	XX	**XX**	**2465**

Tabela 3: Média dos Resultados PSO-Paralelo

Instâncias	Estratégia 1	gap(%)	Estratégia 2	gap(%)	Estratégia 3	gap(%)	Estratégia 4	gap(%)	Ótimo
ftv35	1508,1	2,38	1498,2	1,71	1491,5	1,26	1488,1	1,03	1473
p43	5625,5	0,10	5624,1	0,07	5625,3	0,09	5622,5	0,04	5620
brazil58	25928,2	2,10	25777,4	1,51	26138,1	2,93	26052,7	2,59	25395
ftv70	2062,8	5,78	2044,8	4,86	2080,3	6,68	2011,7	3,16	1950
kro124	38344,9	5,84	37692,9	4,04	38661,1	6,71	38230,1	5,52	36230
ftv170	3093,5	12,29	3066,9	11,32	3117,5	13,16	2989,3	8,50	2755
rbg323	1328,8	0,21	1328,3	0,17	1328,5	0,19	1328,3	0,17	1326
rbg403	2469,1	0,17	2469,4	0,18	2471,2	0,25	2467,2	0,09	2465
pr299	51555,4	6,98	51348,32	6,55	51326,03	6,51	50455,15	4,70	48191
pr439	178968,7	66,92	147942,14	37,98	140848,5	31,37	137361,6	28,12	107217
pr1002	2051478,75	691,94	1501262,7	479,54	312942,33	20,81	567993	119,26	259045

Tabela 4: Melhores Resultados PSO-Paralelo

Instâncias	Estratégia 1	gap(%)	Estratégia 2	gap(%)	Estratégia 3	gap(%)	Estratégia 4	gap(%)	Ótimo
ftv35	1492	1,29	1475	0,14	1473	0,00	1473	0,00	1473
p43	5622	0,04	5621	0,02	5621	0,02	5620	0,00	5620
brazil58	25400	0,02	25445	0,20	25450	0,22	25400	0,02	25395
ftv70	1968	0,92	1991	2,10	1959	0,46	1959	0,46	1950
kro124	37534	3,60	36879	1,79	37485	3,46	37249	2,81	36230
ftv170	2967	7,70	2920	5,99	2917	5,88	2917	5,88	2755
rbg323	1327	0,08	1326	0,00	1326	0,00	1326	0,00	1326
rbg403	2465	0,00	2465	0,00	2465	0,00	2465	0,00	2465
pr299	49359	2,42	49279,8	2,26	49578	2,88	49578	2,88	48191
pr439	163322	52,33	121447	13,27	115512	7,74	114810	7,08	107217
pr1002	1050480	305,52	929126	258,67	278671	7,58	290393	12,10	259045

Capítulo 6

Conclusões

Como conclusão deste trabalho, pode-se afirmar que, está sendo de suma importância o estudo da técnica do PSO aplicado ao problema de Caixeiro Viajante. Os próprios resultados mostram uma melhoria significativa a cada modificação na estrutura e\ou estratégia aplicada trazendo boas expectativas quanto a sua utilização no problema de otimização combinatória analisado, seja em relação ao tempo de execução ou aos valores alcançados. Foi observada a importância da necessidade da diversificação das partículas no enxame, principalmente a partir da troca de partículas entre os bairros, fazendo com que as partículas que forçassem um maior deslocamente das restantes presentes no enxame.

O principal ponto de destaque foi a paralelização do PSO para resolução do problema do caixeiro viajante, ainda pouco explorado na literatura, que apesar das melhorias apresentadas como a diminuição do enxame, da redução do tempo e da aproximação dos resultados obtidos pelos resultados ótimos já previamente conhecidos, ainda não se mostrou tanta eficiência com problemas considerados grandes, a partir de 1500 nós.

Como sugestão, para pesquisas futuras, o PSO poderia ser aplicado em conjunto com outras metaheurísticas, devido a sua fácil parametrização e aplicação de outras buscas locais para verificação do comportamento de resposta do algoritmo. Outra sugestão é aplicar os algoritmos propostos para o PSO nesta dissertação a outros problemas de otimização combinatória, com características populacionais.

Referências

RIBEIRO, C.C. Heuristiques and Metaheuristiques, ISIMA, 2002.

CAMPELO, R. E.; MACULAN, N., Algoritmos e Heurísticas Desenvolvimento e Avaliação de Performance, 1994.

RIBEIRO, C.C. "Metaheuristics and Applications. In Advanced School on Artificial Intelligence", Estoril, Portugal, 1996.

RECHENBERG, I., "Evolution strategy: Optimization of technical systems by means of biological evolution", Chichester, 1981.

KENNEDY ,J.; EBERHART , R. C., Swarm Intelligence, 2001.

CLERC, M "Discrete Particle Swarm Optimization Illustrated by the Traveling Salesman Problem", 29 February 2000, disponível em http://www.mauriceclerc.net.

GUEDES, A. C. B.; LEITE, J. N. F.; ALOISE D. J., Um Algoritmo Genético com Infecção Viral para o Problema do Caixeiro Viajante, 2004.

DANTZIG; FULKERSON, and JOHNSON. *Solution of a Large--Scale Traveling--Salesman Problem*. Journal of the Operations Research Society of America, 1954

BODIN, L.D.; ASSAD, B. L.;BALL, A..Routing and Scheduling of Vehicles and Crew, the State of the Art. Computers & Ops. Res. 10, 69-211.

CHAUNY, F.; HAURIE, A.; WAGNEUR, E.; LOULOU, R.. Punch Operations in a Flexible Manufacturing cell a three-dimensional space-filling curve approach. INFOR 25(1), 28-45, 1987.

REINELT, G.. The Traveling Salesman: Computacional Solutions for the Applications. Lectures Notes in Computer Science 840, Spring-Verlag, Berlim, 1994.

WHITLEY, D.; STARKWHETER, T.; SHANER, D.. The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Recombinations. Handbook of Genetic Algorithms, Edt L. Davis Van Nostrand, 350-372, 1991.

COTTA, C.; ALDANA, J.F.; NEBRO, A.J.; TROYA, J.M., Hybridizing Genetic Algorithms with Branch and Bound Techniques for the Resolution of

The TSP, 1995.

BERTSKAS D.P., Dynamic Programming – Deterministic and stochastic Models, ed. Prentice Hall, Inc . 1987

AARTS E.; KORST, J., *Simulated Annealing and Boltzmann Machine,* John Wiley, 1989

KIRKPATRICK, S.; GELLAT, C.D.; et VECCHI, M.P.. *Optimization by Simulated Annealing*".Science, 220:671-680, 1983.

GLOVER, F.. "*Tabu Search: fundamental and uses*". Rapport Technique, Graduate School ofBusiness, University of Colorado, Boulder, 1996.

PRAIS, M.; RIBEIRO, C.C. "*Parameter variation in GRASP implementations.*(In Proc. of the Third Metaheuristics International Conference, páginas 07 e 08. Angra dos Reis (RJ) –Brasil, 1999.

FEO, T. A.; RESENDE, M. G. C. "*Greedy Randomized Adaptive Search Procedures".*(Kluwer Academic Publishers, Boston), Journal of Global Optimization, 1995.

TOMASSINI M., " *A Survey of Genetic Algorithm*", II Anal Reviews of ComputationalPhysics, World Scientific, 1998

MAURICIO G.C; RESENDE; PANOS M. ;PARDALOS; SANDRA, " *Parallel Metaheuristics for Combinatorial Optimization*", 1999

FORTUNA A.O, " *Introdução ao MPI*", slides

VIANNA L.S, " *Metaheurísticas Seqüenciais e Paralelas Aplicadas a Problemas de Escalonamento e Roteamento", 2002*

SOUZA G. R., " Uma Abordagem por Nuvem de Partículas para Problemas de Otimização Combinatória", 2006

FILHO G. F. S., " Metaheurísticas Seqüenciais e Paralelas para a Configuração de um Serviço de Distribuição de Vídeo Digital ao Vivo", 2008

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo