Capítulo 1

Introdução

Ao longo da história da humanidade, maximizar ganhos e minimizar perdas se mostraram problemas recorrentes e fundamentais para o desenvolvimento científico e tecnológico. O aumento gradativo da complexidade dos casos de otimização estudados resulta na procura incessante de novas soluções e técnicas de busca de resultados nas diversas áreas do saber. Por exemplo, os estudos sobre otimização relacionados às transformações de energia, seus impactos e conseqüências econômicas e ambientais têm sido de vital importância para o desenvolvimento sustentável do mundo atual.

Na Engenharia Nuclear, nos defrontamos com alguns destes problemas desafiantes que incentivam a pesquisa de novas técnicas. Um deles está relacionado à obtenção de energia nuclear, especificamente o da recarga de combustível em usinas nucleares. Em uma operação de recarga, parte do combustível nuclear queimado é retirada do núcleo do reator, sendo substituída por combustível novo, que produzirá reações nucleares com maior probabilidade, comparado ao combustível que permaneceu no núcleo. Isto se deve a uma série de motivos (DUDERSDADT e HAMILTON, 1976), como o aumento da concentração de produtos de fissão nuclear, resultantes das reações, conforme o passar do tempo, influenciando diretamente a distribuição de potência no núcleo. A fim de otimizar a queima são necessários, então, critérios específicos para a alocação dos elementos combustíveis novos, bem como para a reorganização das posições dos antigos.

Um típico reator PWR (*Pressurized Water Reactor*, Reator a Água Pressurizada) (TODREAS e KAZIMI, 1990) pode conter em seu núcleo 193 elementos combustíveis. Estes elementos são grupos de varetas de aproximadamente 3m de altura que contêm as pastilhas de dióxido de urânio enriquecido, o combustível nuclear. Os diferentes arranjos das posições que os elementos antigos e novos ocupam é que vão produzir maior ou menor reação em cadeia, com conseqüências diretas na distribuição de potência e geração de energia térmica, necessária para a produção da energia elétrica. Desta forma, a busca consiste em achar padrões de carregamento (*Loading Pattern*, LP), ou seja, combinações dos elementos combustíveis, que maximizem a vida útil do combustível nuclear, permitindo o aumento do ciclo de queima do combustível, trazendo ganhos operacionais e econômicos relacionados ao funcionamento da usina. Vale ressaltar que a busca por padrões de carregamento está sujeita a restrições relacionadas ao fluxo de nêutrons e à densidade de potência gerada no reator (McFARLANE, 1972), como por exemplo, a obtenção da máxima uniformidade na curva de densidade de potência e a não-existência de picos de potência, ou seja, posições no reator onde a produção de energia local é muito maior do que a média.

Entretanto, para um reator PWR, por exemplo, como o da Usina Nuclear de Angra I, contendo 121 elementos combustíveis, existem aproximadamente 121! permutações possíveis de posicionamento, cálculo que gera um número da ordem de 10^{200}, tornando impossível a avaliação de todas elas. Para redução do espaço de busca, utilizam-se modelagens para o núcleo do reator que o reduzem a 1/4 ou 1/8 do seu formato original, tendo como base sua simetria, como é mostrado no esquema que representa a vista superior de um núcleo de reator a água pressurizada, no apêndice A. E ainda assim, o número de combinações é muito grande. Se todas estas possibilidades tivessem que ser

avaliadas uma por uma, o tempo computacional seria excessivo, tornando inviável o processo.

Além disso, avaliar uma única combinação de elementos combustíveis também tem sua complexidade. Para achar o valor da função objetivo são utilizados códigos de Física de Reatores elaborados segundo as modelagens existentes e que realizam os cálculos numéricos necessários para solucionar os sistemas de equações diferenciais de transporte ou difusão de nêutrons, a fim de determinar, a partir daí, a distribuição de potência, a reatividade, a queima (*burn-up*) e os demais importantes resultados para análises e previsão do funcionamento de um reator nuclear, o que demanda considerável tempo computacional.

Outro ponto importante é que, em um espaço de busca com tantas soluções como o exemplificado, podemos encontrar diversas combinações que tragam bons resultados durante a avaliação. Isto não possibilita, entretanto, afirmar que mesmo a melhor delas seja um ótimo global para todo o espaço de busca. Também não se pode garantir que não será descoberta outra combinação melhor. A existência de muitas combinações ótimas caracteriza a multimodalidade do problema, ou seja, o fato de podermos encontrar uma boa quantidade de soluções ótimas, nenhuma delas sendo necessariamente a solução ótima global.

Fatores como a multimodalidade, o grande número de soluções possíveis, a não-linearidade do problema e o tempo de avaliação da função objetivo dificultam o problema da recarga. Métodos tradicionais de busca e otimização que resolvem problemas em muitas áreas não apresentam eficiência para solucionar este tipo de

problema. Muitos pesquisadores vêm desenvolvendo técnicas de Inteligência Artificial (IA) para a busca de soluções nestas classes de problemas. Vários algoritmos têm sido elaborados e testados para esse fim, como os Algoritmos Genéticos (*Genetic Algorithms*, GAs) (SANTOS, 1998; CHAPOT, 2000), Sistemas de Colônias de Formigas (*Ant Colonization Systems*, AS) (LIMA, 2005) e P-BIL (*Population Based Incremental Learning*) (MACHADO, 1999 e 2005; LIMA, 2000).

O foco deste trabalho é a utilização da técnica de IA (meta-heurística) de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO) (EBERHART e KENNEDY, 1995; PARSOPOULOS e VRAHATIS, 2002) com vistas à otimização da recarga de combustível em um reator nuclear. O PSO apresenta algumas características evolucionárias, assemelhando-se ao GA em alguns pontos. Por exemplo, a busca utilizando uma população de possíveis soluções com as regras de transição entre uma geração e outra envolvendo probabilidades. Filosoficamente, o PSO se diferencia: representa a metáfora do aprendizado social, que leva em conta que o indivíduo pode obter vantagem tanto de sua própria experiência quanto a partir do conhecimento adquirido por outro elemento de seu grupo, como será detalhado adiante. É um método que pode ser facilmente implementado, além de usar operadores aritméticos simples, e que mostrou ser eficiente em vários problemas de otimização.

A técnica de PSO foi apresentada por EBERHART e KENNEDY (1995). Seguindo este modelo, várias aplicações para problemas de otimização em espaços contínuos foram elaboradas. PARSOPOULOS e VRAHATIS (2002) mostram um apanhado da técnica com sua utilização em diversos problemas, tais como Funções Multiobjetivo, Minimax, Programação Linear Inteira (*Integer Programming*) e funções

com ruído e mudança contínua, mostrando a robustez e a versatilidade da técnica. Para espaços de busca discretos, temos algumas contribuições, como as de EBERHART e KENNEDY (1997), que desenvolveram uma versão discreta binária para o algoritmo do PSO onde as partículas apresentam os valores de vetores binários de comprimento n, com as velocidades representando as probabilidades de mudança do valor do *bit*; WANG et al. (2003), que apresentam uma aplicação na solução do Problema do Caixeiro Viajante (*Traveling Salesman Problem*, TSP), com base em Seqüências de Troca (*Swap Sequences*) e Operadores de Troca (*Swap Operators*), mostrando resultados para um TSP contendo 14 cidades; CLERC (2004), com uma forma de adequar o PSO para solucionar um TSP com 17 cidades, usando estratégias específicas; YIN (2004), que adaptou o PSO discreto de KENNEDY e EBERHART para determinar aproximações poligonais ótimas de curvas digitais; e SALMAN et al. (2002), que aplicaram o PSO ao problema de alocação de tarefas (*Task Assignment Problem*, TAP), simplesmente truncando os números reais obtidos, transformando-os em inteiros. Em particular esta última aplicação e o fato de não ser utilizada a parte decimal dos números encontrados trouxeram importantes contribuições para o desenvolvimento da metodologia apresentada neste trabalho. É importante ressaltar que no momento, apesar de ser um algoritmo eficiente para funções contínuas, não existe modelo satisfatório para a versão discreta do PSO.

A procura pela solução do TSP tem muitas correlações com o problema da recarga. Do ponto de vista matemático, ambos os problemas são combinatórios, onde suas possíveis soluções se apresentam sob a forma de conjuntos discretos, correspondendo a um arranjo entre os elementos do problema. Para o TSP, o objetivo é achar a melhor ordem de visitação das cidades por onde o caixeiro deve passar; para a

recarga, o melhor padrão de carregamento, ou seja, a melhor permutação de um certo número de elementos combustíveis em suas posições no núcleo do reator. Como a técnica de busca que for aplicada ao TSP pode ser aplicada à recarga de reatores, já que ambos os problemas são de natureza combinatória, foram escolhidos dois conhecidos TSPs para realização dos testes para os métodos aqui desenvolvidos: o Oliver 30 e o Rykel 48 (TSPLIB, 2005; MACHADO, 1999).

Neste trabalho, a principal adaptação introduzida no PSO para a solução dos TSPs mencionados teve como base o modelo de chaves aleatórias (Random Keys, RK) (BEAN, 1994), utilizado nos GAs. Outras modificações que foram feitas com o objetivo de agilizar o processo de otimização também serão descritas e comentadas.

Enfim, as adaptações e modificações realizadas no algoritmo PSO têm como finalidade possibilitar sua utilização na otimização da recarga de combustível em reatores nucleares. Neste trabalho, procuramos mostrar mais esta aplicação da técnica, utilizando problemas combinatórios de referência na área, com a apresentação dos resultados obtidos.

A fundamentação teórica do PSO é discutida no capítulo 2, com a formulação de seu algoritmo básico e a revisão da técnica de Random Keys.

No capítulo 3, apresentamos formalmente o problema do TSP, proposto para a aplicação do PSO, sua formulação matemática e suas características, que o relacionam com o problema da recarga.

Mostramos no capítulo 4 o PSORK, técnica de otimização que combina PSO e RK, bem como as complementações introduzidas para auxiliar a codificação e decodificação das informações e simulação de fenômenos com o enxame, visando a agilizar o aprendizado do sistema.

O capítulo 5 traz os testes realizados, seus resultados, análises e discussões.

As considerações finais e propostas para próximas etapas de pesquisa são apresentadas no capítulo 6.

Capítulo 2

Fundamentação Teórica

Neste capítulo, descrevemos as idéias que fundamentam o PSO e mostramos seus principais aspectos teóricos, juntamente com sua formulação matemática. Também é feita a revisão das principais características da técnica de Random Keys para codificação e decodificação no TSP.

2.1. Aspectos Teóricos do PSO

O conceito de enxame de partículas por KENNEDY e EBERHART (1995) para otimização de funções contínuas não-lineares e teve como base modelagens para a simulação de grupos sociais simplificados.

Modelos que simulassem comportamentos sociais, como o de um bando de pássaros ou o de um cardume de peixes, estavam sendo estudados em torno de 1990 e serviram como fonte de inspiração para a elaboração da técnica. Entretanto, o interesse dos pesquisadores girava em torno da estética da coreografia e da descoberta de leis que descrevessem a dinâmica do movimento coletivo daqueles animais, que mesmo estando em grande quantidade, apresentam sincronismo em situações de mudança de direção, espalhamento, reagrupamento etc.

Na sociobiologia vêem-se discussões sobre a vantagem competitiva de se realizar aprendizado individual a partir do conhecimento de outros componentes do grupo (WILSON, 1975. Apud EBERHART e KENNEDY, 1995). Esta habilidade pode ser

observada em muitas espécies, inclusive na humana, havendo, entretanto, uma principal diferença. Enquanto os demais animais realizam este aprendizado para funções instintivas como a defesa contra predadores ou a procura de alimentos e parceiros, os seres humanos também o aplicam nos campos da abstração cognitiva. De qualquer forma, esse tipo de aprendizado é fundamental para vantagens competitivas evolucionárias.

Outro dado importante é que os modelos de simulação apresentados na época traziam processos locais como base para o movimento coletivo. Através deles, cada componente conseguiria ajustar suas condições de posição e velocidade em relação aos outros do grupo. Isto proporcionou solo fértil para a proposição do modelo de otimização utilizando um enxame de partículas.

No início da década de 90, foi proposto um novo modelo de movimento com base no aprendizado coletivo (HEPPNER e GRENANDER, 1990. Apud EBERHART e KENNEDY, 1995). Este, porém, apesar de ser similar aos já apresentados, possuía um detalhe que o fazia diferente dos demais. Os pássaros desta modelagem eram atraídos para uma área de pouso. Isto seria possível quando fosse programada uma tendência maior para os pássaros pousarem em vez de permanecerem em vôo. Quando um deles passasse sobre uma determinada área, seria atraído, e os demais, com o aprendizado coletivo e o passar do tempo também se movimentariam com o intuito de aterrissar.

Ora, o que foi observado por KENNEDY e EBERHART, os pioneiros do PSO, é que localizar um "poleiro" e mover-se em direção a ele são procedimentos semelhantes aos que diversas técnicas de otimização procuram fazer (POMEROY, 2004). Só que

neste caso, a capacidade sócio-cognitiva é o fundamento, pois um pássaro que achou uma posição correspondente a uma solução considerada boa sob algum aspecto pode influenciar outros do bando, fazendo com que caminhem em sua direção, havendo sempre a possibilidade de passarem por alguma posição ainda melhor. Isto aumentaria as chances de serem descobertos melhores resultados conforme a evolução do algoritmo.

Algumas adaptações, logicamente, tiveram que ser efetuadas. Tratar cada indivíduo como pássaro ou peixe, de acordo com as analogias e metáforas utilizadas, seria um pouco inadequado. Isto poderia levar a supor que cada "agente" (este sim, um termo mais coerente) seria dotado de alguma espécie de capacidade cognitiva. Passou-se a utilizar então o termo *partícula* para caracterizar os membros de uma população que procura soluções, já que certas características como volume, massa e dimensões dos agentes não são consideradas para a resolução de problemas, nem levadas em conta para o processo de otimização.

Além disto, é importante notar que cada partícula não consegue avaliar sozinha se sua posição é boa ou ruim. É necessário que ela remeta as coordenadas de uma nova posição encontrada por ela para uma função que as avalie quantitativamente, fornecendo um número como resultado correspondente àquela posição. Esta é a chamada *função objetivo* ou *fitness*, que é uma medida de aptidão, ou seja, que conseguirá dizer quão boa é uma posição em relação a uma outra já encontrada. Mesmo porque, na busca de melhores resultados no PSO, é necessário que uma nova posição encontrada seja comparada tanto com a melhor posição individual, obtida até o momento pela própria partícula, quanto com a melhor posição global, levando em consideração a contribuição

da partícula que encontrou a melhor posição de todo o enxame. Efetivamente, as partículas não executam estas e outras tarefas, de modo que é mais adequado caracterizá-las como "caçadoras" de solução, sem, no entanto, atribuir-lhes capacidades individualmente cognitivas, já que não possuem cérebros individualizados.

Uma partícula é, na verdade, a representação de um objeto que se move ao longo do espaço de busca impulsionado por comparações entre sua posição atual, sua melhor posição atingida até o momento e a posição obtida pela melhor partícula do enxame. Estes termos, devidamente equacionados é que vão proporcionar o cálculo de suas novas posições e velocidades a cada iteração, ou seja, vão proporcionar, enfim, o movimento da partícula. Cada uma das partículas realiza um balanço entre suas capacidades de exploração e de tirar proveito da exploração realizada pelas outras. A exploração é a capacidade de procurar uma boa solução individualmente. Tirar vantagem do sucesso obtido por outro indivíduo é que estaria relacionado ao comportamento de aprendizado social ou coletivo citado anteriormente.

O balanceamento destas características pode fazer grande diferença em um processo de otimização. Pouca capacidade de exploração pode significar convergência prematura para um resultado não necessariamente ótimo global. Pouca capacidade de obter vantagem do sucesso alheio pode fazer com que o enxame perca a habilidade de encontrar uma boa solução, e neste caso, não convergir para um resultado. Ou seja: o sucesso da busca realizada pelo enxame está condicionado ao balanço entre individualidade e coletividade, justamente os traços que compõem o comportamento social.

Em resumo, o algoritmo PSO teve seu início a partir da simulação de um meio social inspirado basicamente no comportamento de um grupo de pássaros em seu movimento coreografado, com a busca de uma área específica para pouso. Realizadas as devidas adaptações como comparar um poleiro a uma solução ótima, esta forma de representação artificial derivada da biologia consegue obter sucesso em diversos modelos e problemas de otimização. O sucesso do PSO é devido ao fato de que os grupos que possuem os melhores padrões sociais de comportamento tendem a se fortalecerem e, com isto, sobreviverem. Os indivíduos garantem êxito a partir de sua própria experiência, bem como da experiência de outros do grupo. Na próxima seção veremos como estes aspectos são representados matematicamente no algoritmo clássico do PSO.

2.2. Algoritmo Básico do PSO

2.2.1. As Equações Fundamentais do PSO

O algoritmo clássico do PSO, voltado para a otimização de funções contínuas, utiliza vetores do tipo

$$\mathbf{a_j} = (a_1, a_2, ..., a_n) \quad (1)$$

para representar posições e velocidades (vetores deslocamento). No vetor dado, n é o número de dimensões do espaço de busca do problema e j = 1, 2, ..., n.

Para representar a i-ésima partícula de um enxame com P partículas será utilizado o vetor

$$\mathbf{x_{ij}} = (x_{i1}, x_{i2}, \dots , x_{in}), \quad (2)$$

com i = 1, 2, ..., P. A posição de cada partícula representa uma possível solução para o problema, que será avaliada a cada iteração t+1 por uma função objetivo ou *fitness* $f(\mathbf{x_{ij}}^{t+1})$ específica.

O vetor que representa a melhor posição global, obtida pela melhor partícula do enxame é

$$\mathbf{gbest_j} = (\text{gbest}_1, \text{gbest}_2, \ldots, \text{gbest}_n), \tag{3}$$

que apresenta a melhor *fitness* entre todas até então encontradas. E para a melhor posição obtida para cada partícula i temos os vetores

$$\mathbf{pbest_{ij}} = (\text{pbest}_{i1}, \text{pbest}_{i2}, \ldots, \text{pbest}_{in}). \tag{4}$$

Os vetores deslocamento poderiam ser chamados $\mathbf{\Delta x_{ij}}$, mas para simplificação da notação são chamados velocidades $\mathbf{v_{ij}}$ e vão ser representados por:

$$\mathbf{v_{ij}} = (v_{i1}, v_{i2}, \ldots, v_{in}). \tag{5}$$

Vale ressaltar que esta simplificação de notação (KENNEDY e EBERHART, 2001) não traz prejuízo à análise dimensional das fórmulas existentes.

Enfim, para uma iteração t+1, a velocidade e a posição de cada partícula são atualizadas segundo as fórmulas:

$$\mathbf{v_{ij}}^{t+1} = w\mathbf{v_{ij}}^{t} + c_1 r_1{}^{t} (\mathbf{pbest_{ij}} - \mathbf{x_{ij}}^{t}) + c_2 r_2{}^{t} (\mathbf{gbest_{ij}} - \mathbf{x_{ij}}^{t}) \tag{6}$$

$$\text{e} \qquad \mathbf{x_{ij}}^{t+1} = \mathbf{x_{ij}}^{t} + \mathbf{v_{ij}}^{t+1}, \tag{7}$$

onde w é o peso de inércia; c_1 e c_2 são constantes de aceleração; r_1 e r_2 são números aleatórios de uma distribuição de probabilidade uniformemente distribuída, pertencentes ao intervalo [0, 1].

A cada iteração t+1, temos um vetor calculado $\mathbf{x_{ij}^{t+1}}$. Em função deste, é feita a avaliação da *fitness* $f(\mathbf{x_{ij}^{t+1}})$. Dado um problema de minimização, se, em alguma avaliação de fitness realizada em alguma iteração t+1, for obtido

$$f(\mathbf{x_{ij}^{t+1}}) < f(\mathbf{gbest_j}), \tag{8}$$

então o valor de $\mathbf{gbest_j}$ é imediatamente atualizado, assumindo o valor de $\mathbf{x_{ij}^{t+1}}$ em questão.

Da mesma forma, se for obtido

$$f(\mathbf{x_{ij}^{t+1}}) < f(\mathbf{pbest_{ij}}) \tag{9}$$

em algum momento, então o valor de $\mathbf{pbest_j}$ também é atualizado, assumindo o valor de $\mathbf{x_{ij}^{t+1}}$.

Assim são feitas atualizações sucessivas, com as melhores posições individuais e global influenciando diretamente a procura por melhores resultados, como pôde ser observado através das equações (6) e (7).

2.2.2. Interpretação Geométrica das Equações

Podemos também interpretar geometricamente as equações fundamentais do PSO, para um sistema onde n = 2, ou seja, com duas coordenadas x_1 e x_2. Os vetores da equação (7) podem ser representados como na figura 2.1.

O fato de a equação (6) apresentar variáveis randômicas r_1^t e r_2^t selecionadas a cada iteração t, faz com que o vetor $\mathbf{v_{ij}^{t+1}}$ fique restrito a uma determinada área, limitada pelo paralelogramo cujos lados têm as direções dos vetores $\mathbf{pbest_{ij}} - \mathbf{x_{ij}^t}$ e $\mathbf{gbest_j} - \mathbf{x_{ij}^t}$. Tais vetores apontam respectivamente nas direções da melhor posição encontrada pela

partícula i e da melhor posição global. Para ilustrar, vamos considerar inicialmente a equação (6) com um valor w = 0, que anula o primeiro termo de seu segundo membro. Assim passamos a ter:

$$\mathbf{v_{ij}}^{t+1} = c_1 r_1^{\,t}(\mathbf{pbest_{ij}} - \mathbf{x_{ij}}^{t}) + c_2 r_2^{\,t}(\mathbf{gbest_j} - \mathbf{x_{ij}}^{t}). \qquad (10)$$

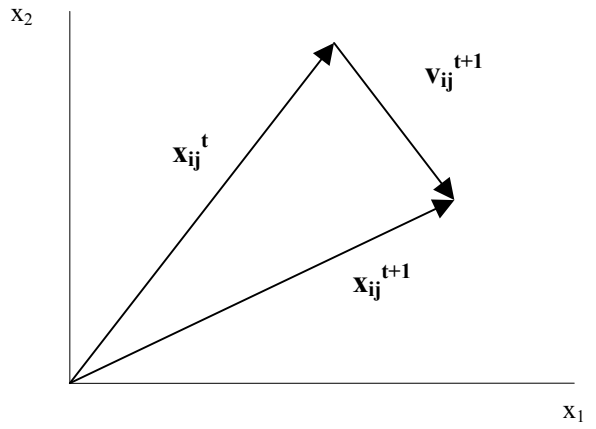


Fig. 2.1: Representação dos vetores da eq. (7) para um espaço vetorial de duas dimensões x_1 e x_2.

Devido aos fatores $r_1^{\,t}$ e $r_2^{\,t}$, o vetor $\mathbf{v_{ij}}^{t+1}$ poderá ser encontrado em qualquer posição na área S^t limitada pelo paralelogramo formado pelos vetores $c_1(\mathbf{pbest_{ij}} - \mathbf{x_{ij}}^{t})$ e $c_2(\mathbf{gbest_j} - \mathbf{x_{ij}}^{t})$, ilustrado pela figura 2.2.

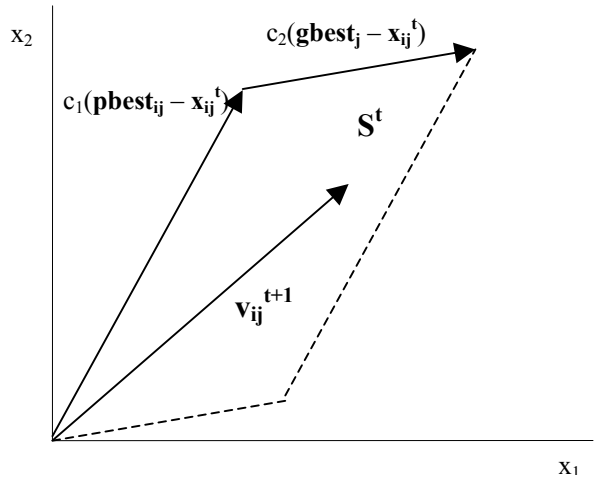


Fig. 2.2: $\mathbf{v_{ij}}^{t+1}$ representa um caso particular para w = 0, podendo ser, no entanto, qualquer vetor compreendido na área S^t delimitada pelo paralelogramo acima, dependendo dos valores $r_1^{\,t}$ e $r_2^{\,t}$.

Para um caso geral, com $w \neq 0$ sendo estipulado e constante ao longo das iterações do algoritmo, a representação é exibida na figura 2.3.

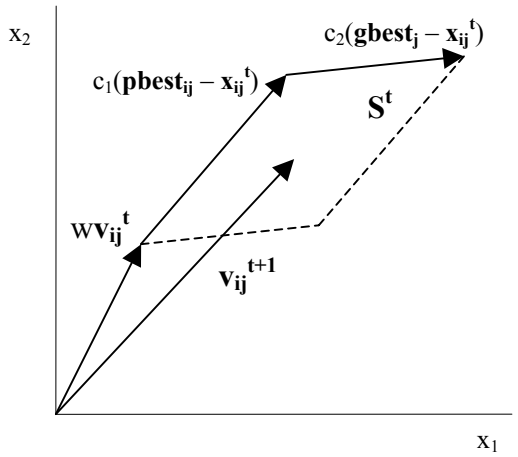


Fig. 2.3: Nesta figura, $\mathbf{v_{ij}^{t+1}}$ representa um caso particular para $w \neq 0$, dependendo dos valores r_1^t e r_2^t.

Enfim, uma re-interpretação da figura 2.1 é mostrada na figura 2.4.

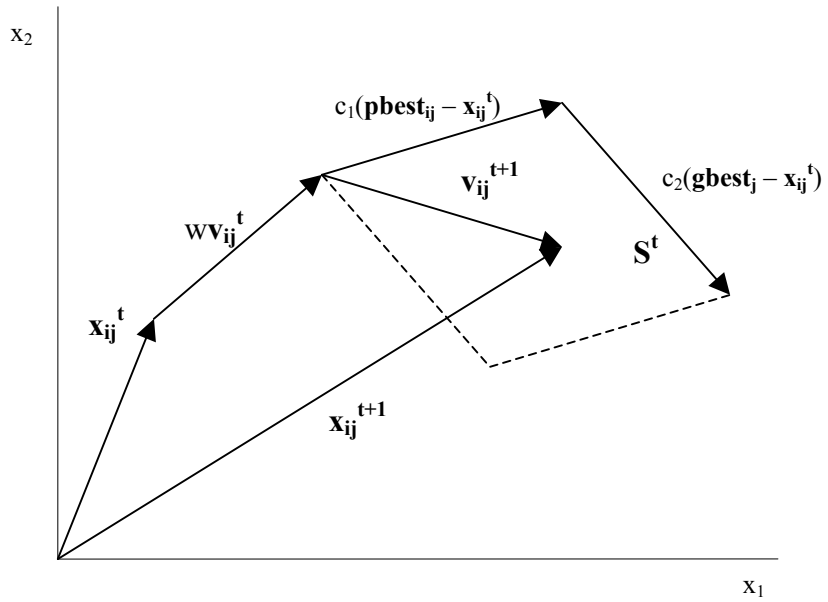


Fig. 2.4: Interpretação geométrica das equações (6) e (7).

Assim obtemos uma interpretação geométrica em duas dimensões para o que acontece a uma partícula i do enxame, quando são atualizadas sua posição e sua velocidade a cada geração através das equações (6) e (7).

Em termos qualitativos, o termo $w\mathbf{v}_{ij}^{t}$ da equação (6) representa a influência de sua velocidade anterior na nova velocidade. Depende, é claro, do peso de inércia w: para valores de w muito altos, o enxame tende a passar direto por boas soluções e acaba ficando perdido, sem convergir; para w muito pequeno, o enxame perde sua habilidade de procurar, levando muitas gerações para haver alguma convergência. Já os fatores $\mathbf{pbest}_{ij} - \mathbf{x}_{ij}^{t}$ e $\mathbf{gbest}_{j} - \mathbf{x}_{ij}^{t}$ indicam para a partícula as direções das melhores posições individuais e globais, com suas *forças de atração* definidas pelos valores $c_1r_1^{t}$ e $c_2r_2^{t}$, respectivamente.

2.2.3. O Algoritmo do PSO

Na figura 2.5, pode-se verificar um exemplo de algoritmo do PSO básico adaptado de SALMAN, AHMAD e AL-MADANI (2002).

Seja *P* o número total de partículas.
Seja *x(i,j)* a posição $\mathbf{x}_{ij}^{t}$ da i-ésima partícula, representando uma solução candidata.
Seja *fitness(i)* a função objetivo $f(\mathbf{x}_{ij}^{t})$.
Seja *v(i,j)* a velocidade $\mathbf{v}_{ij}^{t}$ da partícula i.
Seja *gbest(j)* a melhor posição global $\mathbf{gbest}_{j}$.
Seja *bestfitnessglobal* a melhor fitness, encontrada a partir de *gbest(j)*.
Seja *pbest(i,j)* a melhor posição $\mathbf{pbest}_{ij}$ a i-ésima partícula.
Seja *bestfitnesscadapart(i)* a melhor fitness da partícula i, encontrada a partir de *pbest(i,j)*.

Passo 1 (Inicialização):
- **1.1** Inicializar *x(i,j)* randomicamente.
- **1.2** Inicializar *v(i,j)* randomicamente.
- **1.3** Avaliar *fitness(i)*.
- **1.4** Inicializar *pbest(i,j)* como uma cópia de *x(i,j)*.
- **1.5** Inicializar *gbest(j)* com *x(i,j)* da partícula i de menor fitness.

Passo 2: Repetir até que algum critério seja satisfeito.
- **2.1** Para cada partícula i: se *fitness(i)*≤*bestfitnessglobal* então *gbest(j)* ← *x(i,j)*
- **2.2** Para cada partícula i: se *fitness(i)*≤ *bestfitnesscadapart(i)* então *pbest(i,j)* ← *x(i,j)*
- **2.3** Para cada partícula i: atualizar *x(i,j)*e *v(i,j)* de acordo com as equações (6) e (7).
- **2.4** Avaliar *fitness(i)*.

Fig. 2.5: Algoritmo básico do PSO.

Em suma, pode-se dizer que a estratégia de procura por ótimos no PSO é simples. Usam-se basicamente operadores aritméticos triviais em suas estruturas. A partir desta simplicidade, bons resultados conseguem ser obtidos em diversos tipos de funções contínuas, inclusive as multimodais, como mostram PARSOPOULOS e VRAHATIS (2002), em artigo com minuciosa revisão das aplicações.

A dificuldade, porém, residia na extensão da técnica a problemas combinatórios, onde as soluções candidatas se apresentam como um conjunto de dados discretos. Para superar este obstáculo, associamos ao PSO a técnica de Random Keys, muito utilizada nos Algoritmos Genéticos, cujas características pertinentes ao nosso trabalho serão mostradas na seção seguinte.

2.3 Random Keys

O modelo de Chaves Aleatórias (Random Keys), proposto por BEAN (1994), codifica uma solução com o auxílio de números aleatórios. Os números escolhidos ao acaso em uma distribuição probabilística uniforme em um intervalo [0,1] são utilizados como chaves de ordenamento para formar soluções dentro de um determinado problema.

Seja o problema de agendamento de tarefas em apenas uma máquina (*Single Machine Scheduling Problem,* SMSP) (BEAN, 1994), sem que haja repetição de tarefas. Se for sorteada uma seqüência S_A = (0,39; 0,12; 0,54; 0,98; 0,41) de chaves, o resultado da decodificação será correspondente ao vetor V_A = (2, 1, 5, 3, 4), já que 0,12 é o menor dos números e corresponde à 2ª. posição em S; 0,39 corresponde à primeira posição e

assim por diante. Já para uma seqüência S_B = (0,08; 0,36; 0,15; 0,99; 0,76), o cromossomo seria o vetor V_B = (1, 3, 2, 5, 4).

O vetor V_A assim formado pode ser interpretado como a seqüência de tarefas a serem executadas por uma máquina. Indo mais além, pode ser visto também como a ordem de visitação de um caixeiro viajante a diversas cidades, problema que queremos abordar neste trabalho.

A motivação de se usar este modelo nos Algoritmos Genéticos, onde foi aplicado originalmente por BEAN (1994), é que após um cruzamento entre dois cromossomos, com a utilização de um operador de recombinação (*crossover*) ele garantirá que todos os descendentes (*offspring*) serão representações válidas de soluções. Sejam os vetores V_A = (2, 1, 5, 3, 4) e V_B = (1, 3, 2, 5, 4). Como ilustrado pela figura 2.6, se escolhermos o ponto de crossover como sendo o entre a 2ª. e 3ª. posições, os descendentes seriam V_C = (2, 1, 2, 5, 4) e V_D = (1, 3, 5, 3, 4), que não são soluções válidas, pois no SMSP não há repetição de tarefas para alocação e no TSP todas as cidades têm que ser visitadas, e uma única vez.

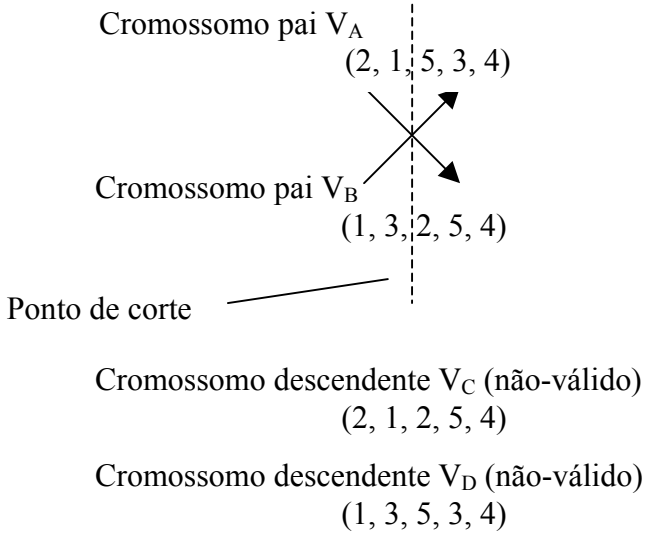
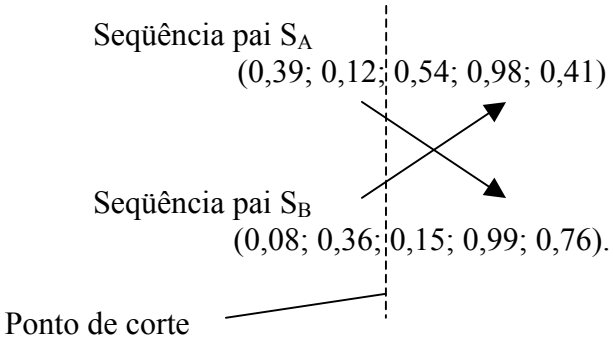


Fig. 2.6: Crossover formando cromossomos descendentes não-válidos.

Contudo, podemos fazer o *crossover* nas codificações que deram origem aos cromossomos em vez de fazermos diretamente nos indivíduos. Por exemplo, em vez do cromossomo V_A, podemos utilizar a seqüência S_A = (0,39; 0,12; 0,54; 0,98; 0,41), que deu origem a V_A. Em vez de utilizar o cromossomo V_B = (1, 3, 2, 5, 4) diretamente, poderíamos utilizar a seqüência S_B = (0,08; 0,36; 0,15; 0,99; 0,76), de cuja decodificação V_B resultou. Com o crossover sendo feito diretamente nas seqüências de chaves, a geração de descendentes válidos fica garantida como mostra a figura 2.7. Assim, dá-se origem a duas novas seqüências e sua decodificação fornece dois indivíduos válidos, o que não ocorreria caso a recombinação fosse feita diretamente sobre os cromossomos.



Seqüência descendente S_C
(0,39; 0,12; 0,15; 0,99; 0,76)
Cromossomo descendente V_C
(2, 3, 1, 5, 4)

Seqüência descendente S_D
(0,08; 0,36; 0,54; 0,98; 0,41)
Cromossomo descendente V_D
(1, 2, 5, 3, 4)

Fig. 2.7: Crossover com RK formando cromossomos descendentes válidos.

Já que todas as cidades no TSP são representadas por números inteiros, um dos obstáculos ao uso do PSO na sua solução seria a formação de vetores-posição reais, devido às fórmulas (6) e (7) apresentarem os fatores $c_1r_1^t$ e $c_2r_2^t$, que são reais. SALMAN et al. (2002), na solução do TAP, utilizaram como critério para eliminar esse obstáculo o truncamento dos reais, para que fossem geradas soluções válidas. Já no presente trabalho, o próprio empecilho – a geração de resultados decimais – foi usado para solucionarmos a questão. Exatamente neste ponto surgiu a oportunidade de utilizar o RK, pela sua capacidade de lidar com vetores de elementos reais em um espaço de busca contínuo. O RK, por fim, foi a técnica que auxiliou o enxame na decodificação, interpretação e aprendizado como será detalhado no capítulo 4.

O capítulo seguinte apresentará o TSP, sua formulação matemática, suas características e classificação dentro da Teoria de Complexidade Computacional.

Capítulo 3

Apresentação do Problema

Este capítulo é destinado à apresentação do Problema do Caixeiro Viajante (*Traveling Salesman Problem*, TSP), que foi escolhido para verificação da técnica desenvolvida neste trabalho por apresentar características semelhantes ao problema da recarga de combustível em um reator nuclear PWR, já que ambos são problemas combinatórios, sendo que no TSP procura-se otimizar a seqüência de cidades sem repetição, e na recarga, otimiza-se o padrão de carregamento (seqüência de elementos combustíveis em posições do núcleo, sem repetição).

É vantajoso utilizar inicialmente o TSP para testes e comparações porque o cálculo da função objetivo no problema da recarga é de alto custo computacional e depende de resultados de códigos de Física de Reatores, enquanto no TSP, o cálculo da avaliação é extremamente trivial e computacionalmente rápido. O TSP, embora possuindo uma formulação simples, é considerado de difícil resolução, destacando-se e servindo como referência entre os problemas dentro da Teoria de Complexidade Computacional. Além disto, trabalhos recentes mostram que técnicas de otimização, quando bem-sucedidas na aplicação ao TSP, levam a resultados satisfatórios quando aplicadas ao problema da recarga de combustível em um reator nuclear (CHAPOT, 2000; LIMA, 2005; MACHADO, 2005). Assim, em um primeiro momento usa-se um TSP de 30 ou 48 cidades, similares conceitualmente ao modelo de simetria utilizado no problema da recarga de combustível em uma usina PWR, do ponto de vista da ordem de grandeza do número de possíveis soluções.

A seguir, apresentaremos o TSP e sua formulação matemática geral. Posteriormente, será realizada uma discussão sobre a Teoria de Complexidade Computacional, para uma classificação mais rigorosa do TSP e apresentação de suas características dentro desta teoria.

3.1. O Problema do Caixeiro Viajante (*Traveling Salesman Problem*, TSP)

O TSP é o mais proeminente dos problemas de otimização combinatória (HOFFMAN e WOLFE, 1985). Os primeiros registros de sua descrição entre os matemáticos datam de 1931-32, apesar de existir um livro alemão de 1832 intitulado "*Der Handlungsreisende, wie er sein soll und was er zu thun hat, um Aufräge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiss zu sein. Von einem alten Commis-Voyageur*" (O Caixeiro-Viajante, como ele deve ser e o que ele deve fazer para ganhar comissões e ser bem-sucedido nos seus negócios. De um caixeiro-viajante veterano).

É um problema cuja formulação geral é simples: dado um número $n \geq 3$ de cidades, também conhecidas como nós, e as distâncias entre elas, determinar o itinerário de menor distância total a ser percorrido, visitando-se todas as cidades uma única vez e terminando o percurso na mesma cidade em que se deu o início. O fato de o caixeiro ter que voltar à cidade de partida torna a solução do problema a busca da melhor permutação cíclica das cidades do enunciado. A figura 3.1 ilustra três possíveis itinerários para um TSP de quatro nós.

Existem diferentes tipos de TSP. É chamado TSP simétrico aquele em que, dadas duas cidades i e j, a distância d_{ij} para ir da cidade i à cidade j é a mesma para o caminho inverso:

$$d_{ij} = d_{ji} \qquad \forall\ i, j = 1, ..., n. \tag{11}$$

As distâncias assim estabelecidas vão resultar em uma matriz quadrada D de distâncias, de ordem n, com os elementos a_{ii} de sua diagonal principal iguais a zero, como mostra a figura 3.2.

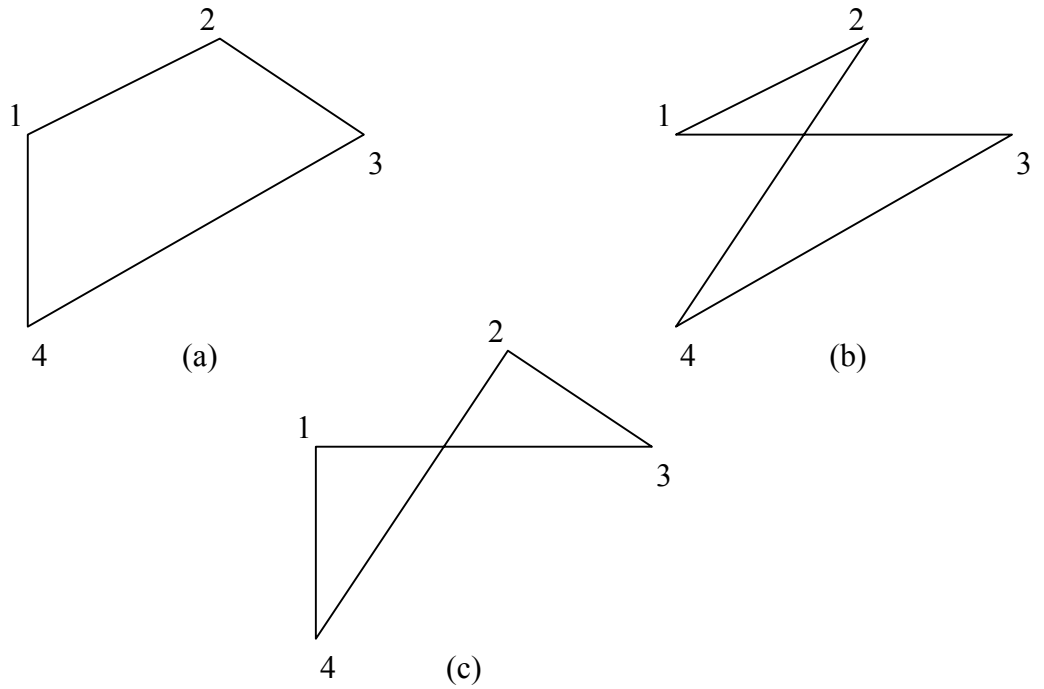


Fig. 3.1: Três possíveis itinerários para um TSP de 4 cidades tomando a cidade 1 como ponto de partida: (a) 1-2-3-4-1, (b) 1-3-4-2-1 e (c) 1-3-2-4-1.

nós	1	2	3	4
1	0	5	11	6
2	5	0	8	10
3	11	8	0	14
4	6	10	14	0

$$D = \begin{Bmatrix} 0 & 5 & 11 & 6 \\ 5 & 0 & 8 & 10 \\ 11 & 8 & 0 & 14 \\ 6 & 10 & 14 & 0 \end{Bmatrix}$$

Fig. 3.2: Quadro de distâncias para um TSP simétrico de 4 cidades e respectiva matriz de distâncias.

Quando existir $d_{ij} \neq d_{ji}$, conforme mostra a figura 3.3, então o TSP é chamado assimétrico.

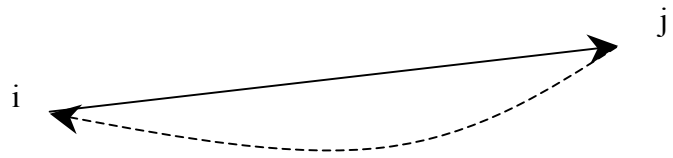


Fig. 3.3: Exemplo de caminhos distintos entre duas cidades de um TSP assimétrico: para ir da cidade i até a cidade j utiliza-se o caminho contínuo; para o caminho inverso, é necessário seguir a linha pontilhada, que possui comprimento diferente.

Para um TSP simétrico com n cidades, temos

$$\frac{(n-1)!}{2} \tag{12}$$

itinerários possíveis. O fatorial de n-1 no numerador corresponde ao cálculo da permutação cíclica. A divisão por 2 decorre do fato de que, para um TSP simétrico, um caminho escolhido resultará na mesma distância percorrida de seu caminho inverso. Um exemplo para três cidades é mostrado na figura 3.4.

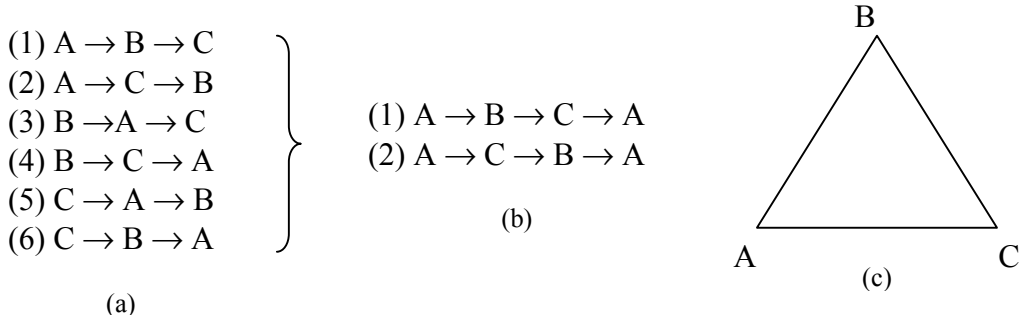


Fig. 3.4: Itinerários equivalentes em um TSP simétrico de n = 3 cidades:
(a) as 6 permutações simples (n!); (b) as duas permutações cíclicas possíveis [(n-1)!];

Um TSP assimétrico com $d_{ij} \neq d_{ji}$ $\forall$ i $\neq$ j terá (n-1)! itinerários diferentes, já que as permutações são cíclicas. Entretanto a distância total para um itinerário não será igual à do caminho inverso.

Os problemas utilizados neste trabalho para testes e comparações foram o TSP simétrico Oliver 30, com 30 nós, que possui cerca de 4,4 x 10^{30} itinerários possíveis e o TSP assimétrico Rykel 48, de 48 cidades, com cerca de 2,5 x 10^{59} permutações cíclicas. A escolha se deve à similaridade em termos de complexidade destes TSPs com o modelo de simetria de 1/8 de núcleo, utilizado para cálculos das modelagens de Física de Reatores, cuja ordem de grandeza do total de possíveis soluções é comparável ao número de soluções possíveis para estes problemas de caixeiro viajante. As matrizes de distâncias para estes problemas e suas soluções são encontradas na TSPLIB (2005).

O TSP é o problema para se encontrar a menor rota passando por todos os nós uma única vez e voltando-se ao nó de onde se partiu ao final da rota. Formalmente, sendo G um grafo completo G = (V, A), onde V é um conjunto de n vértices e A é o conjunto de arcos ou arestas que ligam cada par de cidades i, j $\in$ V, podemos apresentar sua formulação matemática como problema de minimização (BURIOL, 2000):

$$Min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} X_{ij} \tag{13}$$

Sujeito a:

$$X_{ij} \in \{0,1\} \qquad \text{i, j = 1, ..., n} \tag{14}$$

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad \text{j = 1, ..., n} \tag{15}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad \text{i = 1, ..., n} \tag{16}$$

$$\sum_{i \in S} \sum_{j \in S} X_{ij} \leq |S| - 1 \qquad \forall\ S \subset V,\ S \neq \varnothing \tag{17}$$

A variável inteira $X_{ij} = 1$ indica que a cidade j é visitada imediatamente depois da cidade i. Em caso contrário, $X_{ij} = 0$. V é o conjunto {1, 2, ..., n} de cidades do problema e S é um de seus subconjuntos. | S | representa a cardinalidade de S. As restrições (14) e (15) garantem que para cada cidade i existe apenas uma conexão de chegada e uma de saída. A condição (17) garante que não existem sub-rotas, ou seja, rotas que não incluam as n cidades.

Na próxima seção, discutiremos a classificação do TSP na Teoria de Complexidade Computacional.

3.2. Complexidade Computacional

A teoria de complexidade computacional estuda os recursos exigidos durante a solução de um problema, como a memória requerida e o tempo computacional, organizando e classificando os diferentes tipos de problemas em função destes aspectos. Os problemas combinatórios formam um vasto campo de estudo e motivação para esta teoria, compondo uma extensa base de exemplos para se realizarem verificações que permitem a formulação de conceitos e o estabelecimento de paradigmas.

De um modo geral, procura-se na Teoria de Complexidade Computacional (PAPADIMITRIOU e STEIGLITZ, 1982; PAPADIMITRIOU e JOHNSON, 1985) estabelecer definições para a dificuldade de resolver um problema em função do custo

computacional de execução de um algoritmo capaz de resolvê-lo. Assim, podem ser estabelecidas comparações entre os problemas, podendo enquadrá-los em categorias, de acordo com certos critérios. PAPADIMITRIOU e STEIGLITZ (1982) fornecem ampla discussão sobre o assunto e sua relação com os problemas combinatórios.

Com a finalidade de apresentarmos a classificação referente ao TSP nesta teoria, faremos a revisão de alguns importantes conceitos a ela relacionados.

3.2.1. Tempo de Execução

Para estabelecer comparações entre os problemas é necessário estimar o tempo de execução dos algoritmos. Para isto é utilizada a *notação O* (*O-notation*). Os tempos de execução serão do tipo O(f(n)), onde f(n) é uma função de n, com n relacionado aos dados de entrada do problema. Utilizar esta notação significa dizer que existe uma constante c dependente do computador utilizado, tal que o tempo de execução do algoritmo com n dados de entrada é limitado por cf(n). Assim, podem existir algoritmos com tempo de execução O(n!), $O(2^n)$, $O(n^2)$ e assim por diante.

3.2.2. Tempo Polinomial (Polynomial-Time)

Podemos comparar a estimativa de tempo computacional dadas diversas funções f(n) conforme a tabela 3.1, adaptada de PAPADIMITRIOU e JOHNSON (1985). Um algoritmo é considerado bom (*good*) se sua análise pior-caso (*worst case*) utiliza, no máximo, um tempo computacional polinomial em função de n. Por exemplo, para o caso do TSP, PAPADIMITRIOU e JOHNSON (1985) mostram 3 algoritmos

determinísticos para a solução do TSP. Um deles faz gerar todas as soluções possíveis, com avaliação de todas, uma a uma. Isto obviamente pode ser feito, levando a um tempo computacional excessivamente grande conforme o número de cidades de cidades aumenta. Tal algoritmo tem um tempo O(n!) e isto o torna inviável computacionalmente.

Grupo	Função	Valores Aproximados		
Tempo Polinomial	N	10	100	1.000
	n log n	33	664	9.966
	n^3	1.000	1.000.000	10^9
	10^6n^8	10^{14}	10^{22}	10^{30}
Tempo Exponencial	2^n	1.024	$1{,}27x10^{30}$	$1{,}05x10^{301}$
	$n^{\log n}$	2.099	$1{,}93x10^{13}$	$7{,}89x10^{29}$
	n!	3.628.800	10^{158}	$4x10^{2567}$

Tabela 3.1: Comparação entre o crescimento de algumas funções.

Os outros dois algoritmos têm sua base em programação dinâmica (*dynamic programming*), tendo os tempos de execução $O(n^2 2^n)$ e $O(n^3)$, sendo este último um tempo polinomial. Apesar disto, para a Teoria de Complexidade Computacional, a classificação do problema é feita levando em conta o pior caso (PAPADIMITRIOU e JOHNSON, 1985, pp. 42 e 43). Já que, entre os algoritmos determinísticos que solucionam o TSP o pior deles é de tempo exponencial, então, o TSP *é* tempo-exponencial do ponto de vista de sua resolução determinística.

Segundo a teoria, o tempo polinomial garante a eficiência do algoritmo na prática, o que faz a classe incluir até funções que não são polinômios a rigor, mas cujo tempo de

execução é viável, como é o caso da função $n^{\log n}$, no 1º. grupo. Para o 2º. grupo, a inclusão de funções não propriamente exponenciais também pode ocorrer, como acontece com a função n!. Valores de n muito grandes tornam inviável na prática a execução computacional do algoritmo como discutido anteriormente. Esta caracterização, embora ampla, não faz com que a classificação perca consistência ou generalidade, mesmo havendo outros casos específicos, como a função $f(n)=(1,001)^n$, uma função exponencial que não tem crescimento explosivo como as funções do 2º. grupo da tabela. Casos como este não interferem nos resultados da teoria.

3.2.3. Possíveis versões para um mesmo problema de otimização combinatória

Em PAPADIMITRIOU e STEIGLITZ (1982), é estabelecida uma discussão pormenorizada a respeito de três possíveis versões para a formulação de um mesmo problema de otimização, dados a instância (para o TSP, o número de cidades do problema e matriz de distância entre as cidades), o algoritmo para verificar se a instância é uma solução possível e o algoritmo para calcular o custo de uma determinada solução. São elas:

(i) Versão de Otimização: é a formulação na qual o objetivo é determinar a solução possível ótima.

(ii) Versão de Avaliação: cujo objetivo é achar o custo da solução possível ótima.

(iii) Versão de Reconhecimento [ou versão de problema de decisão, segundo PAPADIMITRIOU e JOHNSON (1985)]: neste caso, é dado um valor inteiro L e

procura-se responder à questão "há uma solução possível tal que seu custo seja menor que L?"

Já que a resposta a esta pergunta é *sim* ou *não*, esta última versão também pode ser enquadrada na categoria dos chamados problemas de decisão, tradicionalmente estudados pela Teoria da Computação.

A Teoria da Complexidade Computacional estabelece classes para problemas de decisão. O TSP não é um problema de decisão, mas é possível reformular qualquer problema de otimização a fim de que se torne um deles. A versão de problema de decisão para o TSP tem a seguinte formulação (PAPADIMITRIOU e JOHNSON, 1985):

TSP DECISÃO

INSTÂNCIA: inteiro $n \geq 3$, matriz n x n $C = (c_{ij})$, onde cada c_{ij} é inteiro não negativo e um inteiro $L \geq 0$.

QUESTÃO: Há uma permutação cíclica π dos inteiros de 1 até n tal que

$$\sum_{i=1}^{n} c_{i\pi(i)} \leq L \quad ? \tag{18}$$

Assim, a rigor, a classificação dada pela teoria da Complexidade Computacional é referente à versão de decisão do TSP. Não há, entretanto, prejuízo de qualquer tipo para a busca de sua solução ou aplicabilidade da teoria, pois as três versões são equivalentes em dificuldade, já que a versão de decisão de um problema não é mais difícil que sua versão original.

3.2.4. A classe P (Deterministic Polynomial-Time Problems)

Dizer que um problema de decisão é da classe P significa dizer que trata-se de um problema que pode ser resolvido por um algoritmo determinístico em tempo polinomial, ou seja, um tempo que na prática é considerado eficiente. São exemplos de problemas na classe P: *Minimum Spanning Tree* (Mínima Medida de Árvore), *Path in a Graph* (Trajetória em um Grafo) e *Maximum Matching* (Máxima Correlação).

Segundo PAPADIMITRIOU e JOHNSON (1985), existem "evidências que sugerem fortemente"[1] que a versão de problema de decisão para o TSP não se encontra na classe P, apesar de existirem provas de que outros problemas não estão na referida classe, entretanto não seria possível estabelecer uma "prova rigorosa" para o TSP.

3.2.5 Redução e Transformação Tempo-Polinomial

3.2.5.1. Redução Tempo-Polinomial

É a redução polinomial que permite estabelecer relações entre os diversos níveis de complexidade dos problemas. Dados dois problemas de decisão A e B, diz-se que A é reduzível a B quando o algoritmo destinado a procurar a solução de A utiliza uma ou mais vezes o algoritmo destinado a resolver B na forma de uma subrotina. Além disto, a execução deste sub-algoritmo, considerado um passo único (*single step*), deve ser tempo-polinomial. Isto implica que a subrotina deve ser chamada um número de vezes limitado polinomialmente.

A vantagem de se estabelecer esta redução fica clara quando levado em consideração o seguinte lema (PAPADIMITRIOU e JOHNSON, 1985, p. 49):

[1] Traduzido pelo autor da dissertação.

LEMA

Se há uma redução tempo-polinomial de A para B e há um algoritmo tempo-polinomial para B, então existe um algoritmo tempo-polinomial para A.

Assim, quando se consegue reduzir, por exemplo, um determinado problema a um outro já classificado como P, garante-se que ambos são da mesma classe.

3.2.5.2. Transformação Tempo-Polinomial

A transformação tempo-polinomial é um caso particular da redução tempo-polinomial. Pode ser interpretada quando a subrotina é chamada exatamente uma única vez e no final do algoritmo para o problema A (PAPADIMITRIOU e STEIGLITZ, 1982, p.352). Este conceito será útil para a definição da classe NP.

3.2.6. A Classe NP (Non-deterministic Polynomial-Time problems)

Problemas combinatórios como o TSP, podem ser caracterizados segundo três formulações que, juntas, definem a classe NP. São elas:

(i) Propriedade de certificado sucinto (*succint certificate property*)

Ter um "certificado" é possuir um objeto matemático (matriz, vetor, grafo etc.) solução para um problema que garante uma resposta *sim* a um problema de decisão, a partir de uma determinada instância. No caso do TSP, uma seqüência de cidades cujo comprimento total do itinerário é menor que um inteiro L, de acordo com a versão do TSP para problema de decisão. Um certificado é dito sucinto nos termos da eficiência

do tempo de computação dos seus custos, ou seja, do tempo computacional exigido para a avaliação da seqüência de cidades candidata a resolver o problema.

(ii) Algoritmo não-determinístico

Algoritmos não-determinísticos são aqueles que possuem a poderosa instrução em pseudocódigo

vá para ambas *linha 1, linha 2*.

Com este comando seria possível a solução de um problema em forma de computação paralela, cujo um possível esquema de execução é mostrado na figura 3.5. Segundo o paradigma da programação procedural, com algoritmos estruturados, isto não seria realístico do ponto de vista da compilação e execução do programa-fonte, mas pode-se considerar que a procura de um enxame de partículas (fig. 3.6) é um método não-determinístico, já que os processos com as partículas são representados em paralelo.

Pode-se dizer que um algoritmo não-determinístico resolve um problema em tempo polinomial se (a) para cada instância pode ser obtida uma resposta *sim* com uma prova (verificação) sucinta e (b) se o número de passos usados a partir do primeiro dos ramos até uma resposta *sim* é tempo-polinomial.

Esta classe de problemas de decisão foi introduzida por S. A. COOK (1971). Apud PAPADIMITRIOU e JOHNSON, 1985). Em outras palavras, tratam-se dos problemas cuja verificação das soluções obtidas por algoritmos não-determinísticos pode ser realizada em tempo polinomial. Esta é a chamada classe NP.

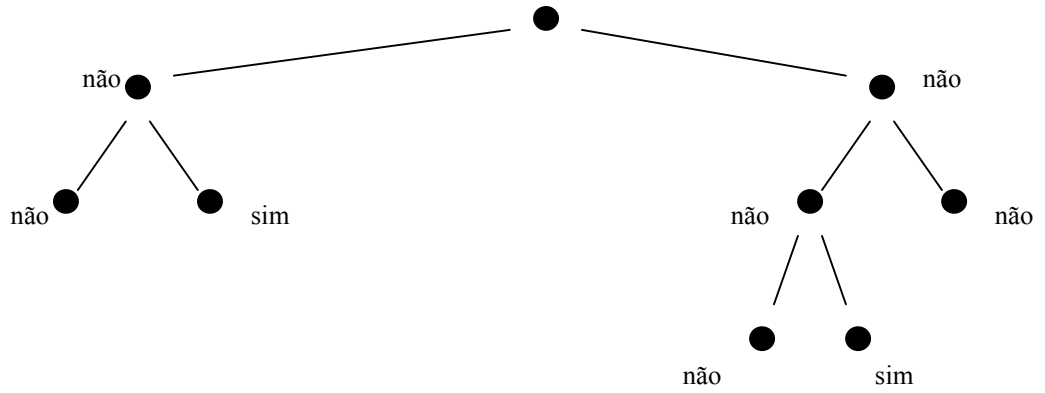


Figura 3.5 [adaptado de PAPADIMITRIOU e JOHNSON (1985), p. 53]: Um possível histórico de execução para um algoritmo não-determinístico.

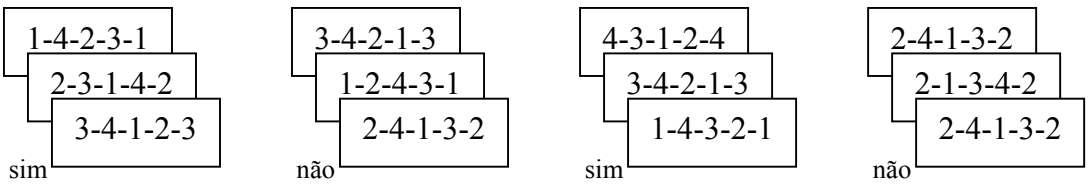


Figura 3.6: Quatro partículas de um enxame trabalhando em paralelo para minimizar um TSP-Decisão simétrico. A seqüência 1-2-3-4-1 ou qualquer permutação cíclica sua foi considerada com custo menor que um L dado

(iii) Transformação tempo-polinomial ao problema da Programação Linear Inteira (*Integer Programming)*

DANTZIG (1960) introduziu esta formulação para problemas que pudessem ser transformados nos problemas de programação linear inteira em tempo polinomial. DANTZIG e vários pesquisadores mostraram que muitos problemas de grafos, programação não-linear e a versão de decisão do TSP e outros problemas combinatórios pertencem a esta classe.

COOK (1971. Apud PAPADIMITRIOU e JOHNSON, 1985) mostrou através de um teorema que as três formulações são equivalentes. PAPADIMITRIOU e JOHNSON

(1985, p. 55) fazem um esboço de sua prova e indicam autores que o provam de maneira rigorosa. Seu enunciado[2] é o seguinte:

> **TEOREMA:** Seja A um problema de decisão. Então as sentenças seguintes são equivalentes:
>
> (A) A tem a propriedade de um certificado sucinto.
>
> (B) $A \in NP$.
>
> (C) A é transformável ao problema da programação linear inteira em tempo polinomial.

Com esta equivalência de enunciados, a classe NP mostra-se consistente e sobretudo importante. Além disto, segundo PAPADIMITRIOU e JOHNSON (1985), qualquer algoritmo determinístico é um caso particular de um não-determinístico ("aqueles algoritmos não determinísticos que não usam a instrução **vá para ambas**"[2]). Desta forma, a classe NP contém a classe P.

Com respeito ao TSP-decisão, ele faz parte do grupo referente à sentença (C) do teorema acima. Já que as três afirmativas são equivalentes, pode-se dizer que o TSP-decisão é NP.

3.2.7. Problemas NP-Completos

A inferência de que o TSP-decisão é intratável, ou seja, que não pode ser resolvido em tempo polinomial, é fortemente aceita na complexidade computacional. Todavia, para provar isto de modo definitivo, seria necessário então provar que $P \neq NP$, o que muitos pesquisadores vêm tentando fazer.

[2] Traduzido pelo autor da dissertação.

Embora a discussão sobre a prova da igualdade ou não dos conjuntos NP e P fuja ao escopo deste trabalho, é importante citar que uma importante classe surge desta discussão. Para tentar chegar a alguma prova conclusiva, é útil definir uma sub-classe de NP que contenha problemas aos quais todos os outros NP possam ser nele transformados. Esta classe que particularmente nos interessa é a dos problemas NP-Completos, subconjunto de NP na qual o TSP-decisão se insere. Uma definição mais formal é dada a seguir:

DEFINIÇÃO (Teorema de COOK): Um problema A é NP-Completo se e somente se:

(A) $A \in NP$

(B) Todos os outros problemas em NP transformam-se polinomialmente em A.

Devido à transitividade da transformação polinomial é possível estabelecer uma árvore de problemas NP-Completos todos transformáveis uns nos outros, com mais de 300 exemplos (GAREY e JOHNSON, 1979. Apud PAPADIMITRIOU e JOHNSON, 1985) como a programação linear inteira, contentabilidade (*satisfiability*) e o TSP-decisão.

3.2.8. Problemas NP-Difíceis (*NP-Hard*)

Tecnicamente, a versão de otimização do TSP não pode pertencer à árvore dos NP-Completos, já que ela não é um problema de decisão, embora seja comparável em complexidade ao TSP-Decisão.

Para estes e outros problemas é definida a classe NP-Difícil. Ela abrange:

(A) problemas de decisão que satisfazem o item (B) da definição de NP-Completo, ou seja, que são transformáveis em outros NP-Completos, mas que não se encontram na classe NP;

(B) problemas NP que são *reduzíveis* a problemas NP-Completos, mas não necessariamente *transformáveis*; e, finalmente,

(C) Problemas que só podem ser resolvidos se P = NP, mas não entram na árvore dos NP-Completos, apesar de serem tão intratáveis quanto qualquer um do tipo NP.

Por esta última razão, pode-se considerar, afinal, que o TSP-otimização, ou seja, o TSP original é do tipo NP-Difícil.

Sendo assim, buscar soluções para um TSP é tratar de um problema peculiar e de referência na área da Computação. Sua utilização é um primeiro passo para validação de qualquer método de otimização criado para problemas conhecidos como NP. A partir daí, os procedimentos podem ser adaptados com a finalidade de otimizar problemas semelhantes a ele em complexidade, como é o caso do problema de otimização da recarga de combustível do reator nuclear, que também pode ser considerado NP-difícil, seguindo raciocínio análogo ao mostrado para classificação do TSP. Por isto, o TSP é o problema utilizado para testes e comparações neste trabalho, sendo um primeiro passo para que, posteriormente, em trabalhos futuros, o método aqui apresentado seja usado em testes para a otimização da recarga. No próximo capítulo, será mostrado o desenvolvimento dos procedimentos que levaram à elaboração da técnica de PSORK, o PSO associado ao RK.

Capítulo 4

Apresentação do PSORK

Este capítulo mostra como foi desenvolvida a técnica híbrida PSORK. Mostraremos no item 4.1 algumas tentativas que conduziram as linhas de raciocínio até as abordagens que proporcionaram a efetiva otimização dos TSPs Oliver 30 e Rykel 48. No item 4.2 apresentaremos o PSORK aplicado ao Oliver 30. No item 4.3, será mostrado o PSORK aplicado ao Rykel 48, para *enxames reduzidos* e *enxames expandidos*, além de serem analisadas algumas sub-rotinas que permitiram a agilização do processo de otimização.

4.1. Procurando um critério para eliminar repetições

SALMAN, AHMAD e AL-MADANI (2002) aplicaram o PSO à otimização do Problema de Alocação de Tarefas (*Task Assignment Problem*, TAP) com o truncamento dos números do vetor de posição das partículas a fim de obter soluções possíveis na otimização discreta. A figura 4.1 mostra um possível exemplo de evolução de uma partícula de uma geração para outra. No caso do TAP cada solução possível contém as tarefas que seriam executadas pelos processadores, podendo haver repetições (fig. 4.1a). Em seguida, sendo realizado o cálculo das velocidades e a atualização do vetor de posição das partículas segundo as fórmulas (6) e (7), respectivamente, será gerada uma solução não-válida, já que todas as tarefas são representadas por números inteiros (fig. 4.1b). O truncamento é então realizado e uma solução válida (fig. 4.1c) pode ser avaliada pela *fitness*.

O trabalho ora desenvolvido pelos autores supracitados forneceu a base para aplicação do algoritmo do PSO a problemas combinatórios. Entretanto, para o TSP, não seria possível utilizar o truncamento, visto que a repetição de cidades invalida suas soluções. A modelagem precisaria agora incluir um critério que eliminasse a repetição de cidades.

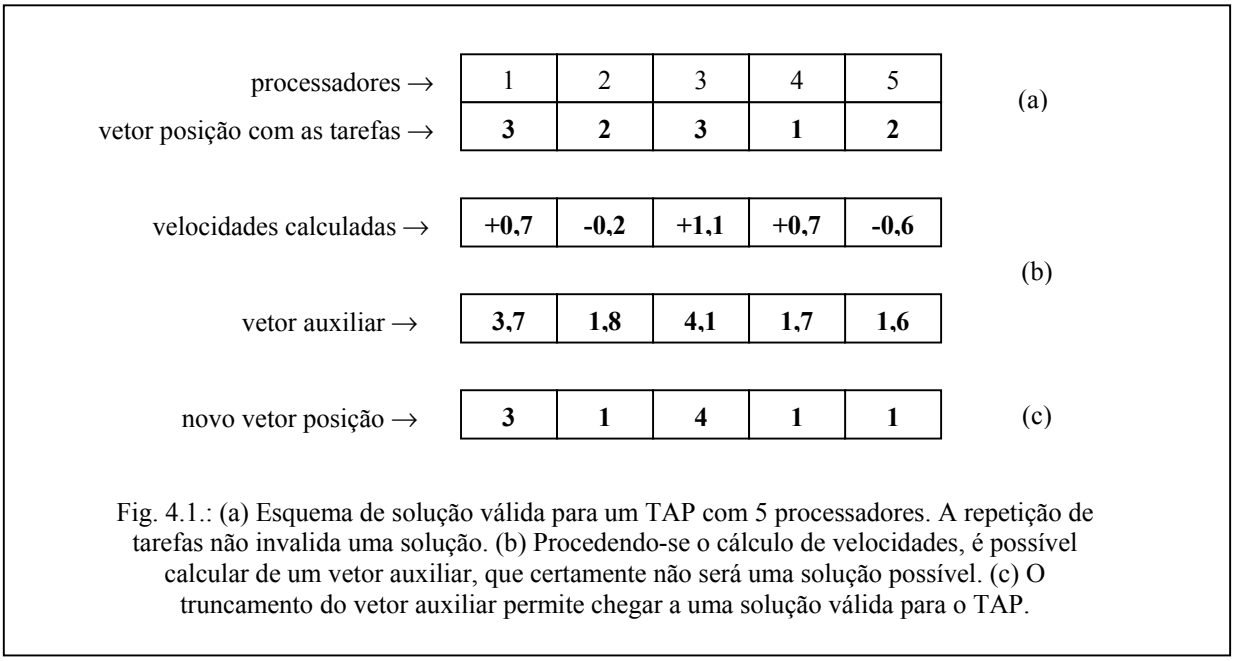


Fig. 4.1.: (a) Esquema de solução válida para um TAP com 5 processadores. A repetição de tarefas não invalida uma solução. (b) Procedendo-se o cálculo de velocidades, é possível calcular de um vetor auxiliar, que certamente não será uma solução possível. (c) O truncamento do vetor auxiliar permite chegar a uma solução válida para o TAP.

Uma primeira tentativa para o TSP foi feita com um vetor que continha as cidades que não seriam visitadas. Por exemplo, o vetor da figura 4.1c daria origem ao vetor **cidades_vagas** = **(**2, 5, 0, 0, 0**)**. Já um vetor posição **(**4, 4, 4, 4, 4**)** teria um vetor de cidades não-visitadas **(**1, 2, 3, 5, 0**)**. Foi então elaborado um algoritmo que verificava a parte decimal dos elementos do vetor auxiliar para cada cidade que estivesse sendo repetida. Para eliminar a repetição do novo vetor posição no caso da figura 4.1c, compariam-se os reais 1,8; 1,7 e 1,6 do vetor auxiliar. O menor deles seria truncado e os seguintes, por ordem crescente, iriam recebendo os elementos do vetor **cidades_vagas**. A figura 4.2 mostra o qual seria o resultado final em um exemplo. Seria um critério razoável, não fosse o fato de, neste procedimento, estarmos alterando

sensivelmente a estrutura da partícula, cancelando de maneira radical todo o aprendizado que a mesma estivesse realizando. Desta forma não haveriam registros que acumulassem aprendizado ao longo das gerações. Apesar de este modelo resolver um TSP de 5 cidades que foi utilizado para teste, mudaria consideravelmente a configuração de uma partícula em um TSP de muitas cidades. Um exemplo disto, para 10 cidades, com a cidade 1 sendo repetida nas posições 2 e 3, seria o vetor auxiliar **(**3, 1, 1, 2, 5, 7, 4, 6, 10, 8**)**. Como a cidade não visitada seria a de número 9, de acordo com este critério, tal cidade seria alocada nas posições 2 ou 3. No entanto, a partícula realizou algum aprendizado ao longo das gerações que a levou àquela configuração, com números baixos de cidades nas primeiras posições. Não se justifica uma cidade que nas primeiras posições seria representada por um número próximo de 1 propiciar um salto como este, e a partícula ser representada com a cidade de número 9 naquelas posições. Assim, todo o aprendizado da partícula estaria sendo desconsiderado. Logo, este método não pode ser utilizado para um TSP com um número maior de cidades.

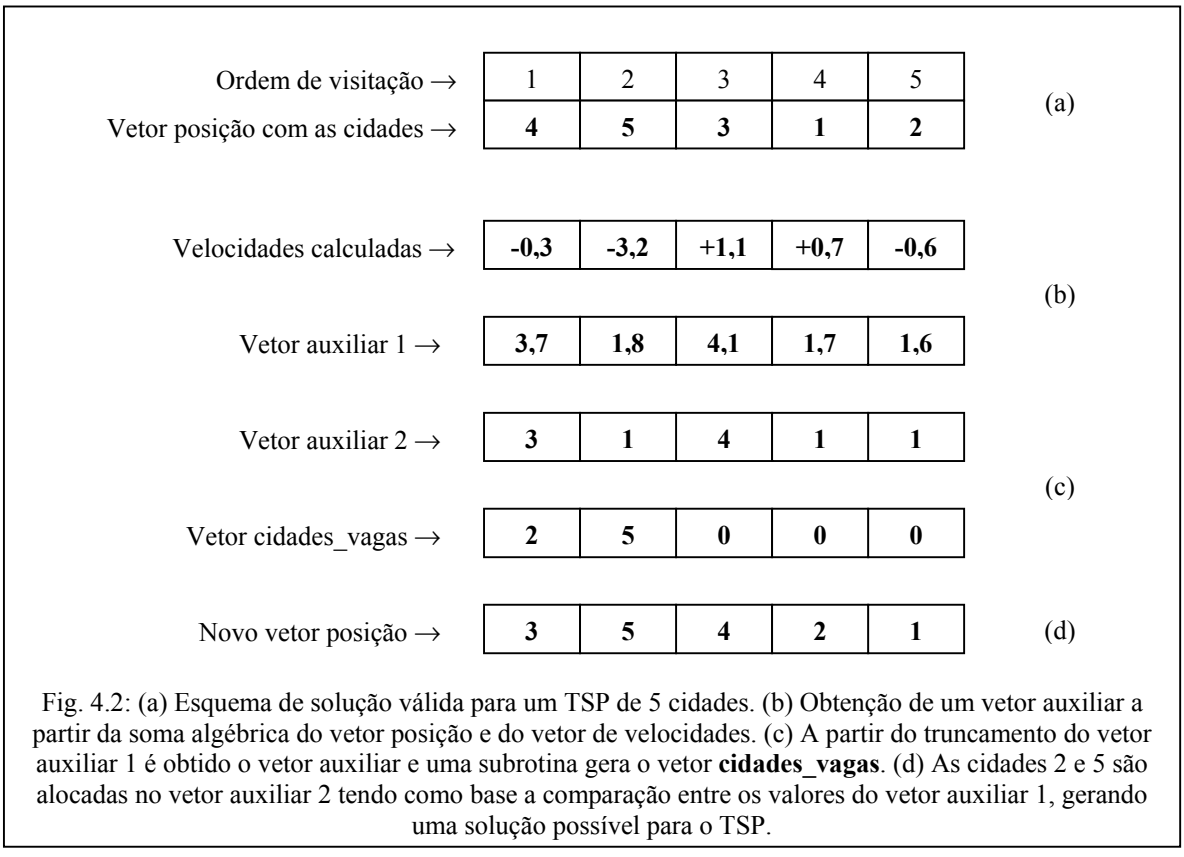


Fig. 4.2: (a) Esquema de solução válida para um TSP de 5 cidades. (b) Obtenção de um vetor auxiliar a partir da soma algébrica do vetor posição e do vetor de velocidades. (c) A partir do truncamento do vetor auxiliar 1 é obtido o vetor auxiliar e uma subrotina gera o vetor **cidades_vagas**. (d) As cidades 2 e 5 são alocadas no vetor auxiliar 2 tendo como base a comparação entre os valores do vetor auxiliar 1, gerando uma solução possível para o TSP.

A partir deste ponto é que se iniciou a procura de um outro método que acumulasse aprendizado ao longo das gerações, que permitisse a decodificação das partes decimais e sua contribuição para o aprendizado da partícula para que pudesse ser aplicado na otimização do TSP Oliver 30.

4.2. Otimização do Oliver 30

Na busca de uma maneira de acumular o conhecimento obtido pela partícula é que se pensou em utilizar o vetor de posições para tornar possível esta tarefa. A cada geração este vetor seria modificado pela velocidade como no método de SALMAN et al. (2002), mas não haveria truncamento. A avaliação realizada pela fitness não seria calculada diretamente em função do resultado dele, mas a partir de um vetor auxiliar, segundo o método RK apresentado na seção 2.3, como pode ser observado na figura 4.3.

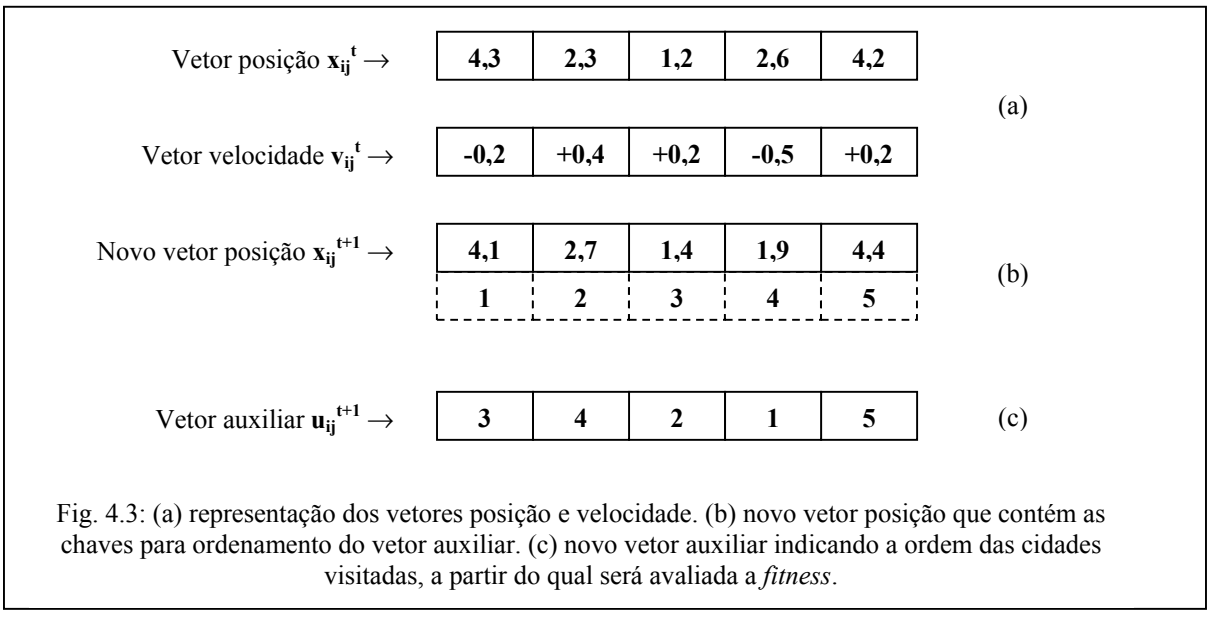


Fig. 4.3: (a) representação dos vetores posição e velocidade. (b) novo vetor posição que contém as chaves para ordenamento do vetor auxiliar. (c) novo vetor auxiliar indicando a ordem das cidades visitadas, a partir do qual será avaliada a *fitness*.

Uma interpretação para o processo que acontece durante a atualização do vetor de posições a partir do vetor de velocidades pode se obtida com base na análise do cruzamento com RK mostrado na figura 2.7. Vamos considerar as seqüências originais S_A = (0,39; 0,12; 0,54; 0,98; 0,41) e S_B = (0,08; 0,36; 0,15; 0,99; 0,76), que dão origem à seqüência descendente S_C = (0,39; 0,12; 0,15; 0,99; 0,76), como no exemplo dado. Sobre esta transição, podemos afirmar que S_A busca a configuração de S_B modificando os elementos situados à direita do ponto de corte, ou seja, do 3º. elemento em diante. Isto é equivalente a somarmos um vetor de incrementos (0,00; 0,00; -0,42; 0,01; 0,35) ao vetor S_A, conforme mostra a figura 4.4. Assim, os elementos do vetor de incrementos são nulos até o ponto de cruzamento, para que seja mantida alguma configuração de A. Após o ponto de corte, as coordenadas são formadas pela diferença entre os elementos da outra seqüência pai S_B e os elementos correspondentes da seqüência S_A. Aplicando este ponto de vista ao PSO, podemos afirmar que o que acontece com o vetor de posições quando a ele é somado o vetor de velocidades também é uma operação de mudança de configuração, a partir da eq. (6). Tal processo seria uma espécie de *hiper-cruzamento*, pois trata-se de uma busca de configuração em cada elemento do vetor. Não seria um processo realizado na direção de uma partícula apenas, com a qual se fez o cruzamento, em um ponto de corte específico, mas sim direcionando todos os elementos do PSO em virtude de informações obtidas pelo vetor de velocidades. Esta é mais uma analogia que pode ser estabelecida entre o PSO e o GA, desta vez do ponto de vista da mudança de configuração e talvez seja a partir daí que a aplicação do RK se torne eficiente.

Seqüência pai $S_A \rightarrow$	**0,39**	**0,12**	**0,54**	**0,98**	**0,41**
Vetor com incrementos $\rightarrow$	**0**	**0**	**-0,39**	**0,01**	**0,35**
Seqüência descendente $S_C \rightarrow$	**0,39**	**0,12**	**0,15**	**0,99**	**0,76**

Fig. 4.4: Uma outra interpretação para o *crossover* com RK. É possível estabelecer uma analogia entre este processo e a soma de velocidades no PSO.

O pseudocódigo do algoritmo utilizado para resolver o Oliver 30 pode ser visto na figura 4.5. É importante destacar que, neste caso, as comparações da equação da velocidade (6) possuem uma diferente interpretação. Na otimização do TSP, $\mathbf{u_{ij}}$ está representando o vetor auxiliar da figura 4.3c, um vetor com números inteiros de 1 a 30 representando uma solução possível, como mencionado anteriormente. A melhor posição individual $\mathbf{pbest_{ij}}$ e a melhor posição global $\mathbf{gbest_j}$ são obtidas a partir dos vetores auxiliares $\mathbf{u_{ij}}$ de melhores avaliações e não do vetor de posições $\mathbf{x_{ij}}$ que contém os números reais (fig. 4.3a). O vetor auxiliar de menor *fitness* em uma geração torna-se a melhor posição global. Ou seja, as comparações existentes na eq. (6) entre a posição atual da partícula sua melhor posição e a melhor posição do enxame, ambos vetores de números reais, são feitas, para o Oliver 30, com base nos itinerários possíveis, todos vetores de números inteiros, apesar de o resultado final da velocidade não ser inteiro. No entanto, a alteração da posição segundo a equação (7) é feita no vetor $\mathbf{x_{ij}^t}$ de chaves reais, sendo $\mathbf{x_{ij}^{t+1}}$ uma nova codificação para ser usada no RK, que vai gerar $\mathbf{u_{ij}^{t+1}}$, que, por sua vez, será utilizado para os cálculos da função de avaliação, havendo em seguida a repetição de todo o processo. Enfim, para a otimização do Oliver 30, foram utilizadas as equações (6) e (7), sendo $\mathbf{pbest_{ij}}$ e $\mathbf{gbest_{ij}}$ obtidas a partir dos $\mathbf{u_{ij}}$ de melhores avaliações e $\mathbf{x_{ij}^t}$ o vetor de posições real que conterá as chaves aleatórias para geração

dos $\mathbf{u_{ij}^t}$, que serão as soluções possíveis (vetor auxiliar com inteiros de 1 a 30, indicando a seqüência de visitação das cidades).

Vale notar que, para a otimização do Oliver 30, também foi implementado um procedimento destinado a manter a diversidade do grupo, que acabou proporcionando agilização do processo. Trata-se da sub-rotina *mata_abelhas* (item 2.1 da fig. 4.5), onde uma partícula é escolhida ao acaso a cada geração, sendo reinicializada aleatoriamente. Assim como no caso da sub-rotina *mata_abelhas*, outra tentativa de estabelecer ganhos no tempo de execução também foi realizada para o Oliver 30. Tentou-se colocar os itens 2.2 a 2.7 da fig. 4.5 em um único laço para que qualquer modificação na melhor posição global em uma geração atuasse junto às partículas avaliadas posteriormente, da mesma geração. Entretanto, isto causou demora considerável na convergência quando foram usadas as mesmas constantes que permitiram achar o mínimo do problema com o algoritmo da figura 4.5.

Em síntese, a principal adaptação na filosofia do algoritmo do PSO, que antes apresentava resultados satisfatórios apenas para otimização de funções contínuas, reside na interpretação do vetor de posições. Na estrutura aqui apresentada, o vetor de posições das partículas não representa as cidades do TSP em si, mas é utilizado como chave para decodificação das informações apreendidas ao longo das gerações. O vetor de posições não precisa, desta forma, ter seus elementos truncados ou arredondados. Através do RK, suas informações são passadas ao vetor auxiliar, este sim representando as cidades, repassando a informação à função objetivo para avaliação. Outra maneira de executar o PSO no processo de otimização também foi desenvolvida e implementada no caso do

TSP assimétrico Rykel 48, permitindo êxito em sua otimização, como veremos no item a seguir.

Seja *P* o número total de partículas.
Seja *x(i,j)* a posição $\mathbf{x}_{ij}^{t}$ da i-ésima partícula, vetor de reais.
Seja *vetor_aux(i,j)* um vetor auxiliar $\mathbf{u}_{ij}^{t}$ para a i-ésima partícula, representando uma solução candidata, um vetor de inteiros.
Seja *fitness(i)* a função objetivo $f(\mathbf{u}_{ij}^{t})$.
Seja *v(i,j)* a velocidade $\mathbf{v}_{ij}^{t}$ da partícula i.
Seja *gbest(j)* o vetor $\mathbf{gbest}_{j}$, o *vetor_aux(i,j)* de menor *fitness* do enxame.
Seja *bestfitnessglobal* a melhor fitness, encontrada a partir de *gbest(j)*.
Seja *pbest(i,j)* o vetor $\mathbf{pbest}_{ij}$, o *vetor_aux(i,j)* de menor *fitness* da i-ésima partícula.
Seja *bestfitnesscadapart(i)* a melhor fitness da partícula i, encontrada a partir de *pbest(i,j)*.

Passo 1 (Inicialização):

1.6 Inicializar *x(i,j)* e *vetor_aux(i,j)* randomicamente.
1.7 Inicializar *v(i,j)* randomicamente.
1.8 Avaliar *fitness(i)*.
1.9 Inicializar *pbest(i,j)*como uma cópia de *vetor_aux(i,j)*.
1.10 Inicializar *gbest(j)* com *vetor_aux(i,j)* da partícula i de menor fitness.

Passo 2: Repetir até que algum critério seja satisfeito.

2.1 Reinicializar aleatoriamente uma partícula escolhida ao acaso (sub-rotina *mata_abelhas*).
2.2 Para cada partícula i: calcular a velocidade *v(i,j)* de acordo com a equação (6).
2.3 Para cada partícula i: atualizar *x(i,j)* de acordo com a equação (7).
2.4 Para cada partícula i: ordenar *vetor_aux(i,j)* usando RK, com as chaves fornecidas por *x(i,j)*.
2.5 Avaliar *fitness(i)* em função de *vetor_aux(i,j).*
2.6 Para cada partícula i: se *fitness(i)*≤ *bestfitnesscadapart(i)* então *pbest(i,j)* ← *vetor_aux(i,j)*
2.7 Para cada partícula i: se *fitness(i)*≤*bestfitnessglobal* então *gbest(j)* ← *vetor_aux(i,j)*

Fig. 4.5: Algoritmo do PSO para otimização do Oliver 30.

4.3. Otimização do Rykel 48

Comparando-se ao Oliver 30, a dificuldade em otimizar o TSP Rykel 48 é maior. Este é um TSP assimétrico, possuindo na ordem de $2{,}59.10^{59}$ possibilidades de itinerários diferentes, ou seja, aproximadamente $5{,}85.10^{28}$ vezes o número de soluções possíveis do Oliver 30. É importante salientar que a abordagem apresentada para o Rykel foi inicialmente diferente da utilizada para o Oliver 30, com algumas subrotinas que auxiliavam no processo de convergência, para enxames contendo em torno de 100 partículas, como será mostrado no item 4.3.1. No item 4.3.2 será mostrado como eliminamos tais subrotinas quando utilizamos enxames de 500 e 1000 partículas.

4.3.1. Otimização do Rykel 48 com Enxames Reduzidos

Neste fase do trabalho, foram utilizados enxames contendo aproximadamente 100 partículas, o que convencionamos chamar *enxames* reduzidos. Nesta etapa, o primeiro mecanismo que agilizou sensivelmente a busca foi inicializar o vetor de posições aleatoriamente no intervalo real (0,48), como mostra o item 1.1 da figura 4.6. No caso do Oliver 30, o vetor de posições era inicializado no intervalo (0,1). Este procedimento foi elaborado porque as constantes que permitiam uma busca adequada, sem convergência prematura também tornavam o processo muito lento em seu início. Isto motivou a inicialização do vetor de posições com reais contidos ao longo do intervalo (0, 48), ou seja, com elementos de ordem equivalente à do número de cidades do problema e isto permitiu uma exploração inicial rápida. O pseudocódigo é exibido abaixo, na figura 4.6.

Procurando outros processos que garantissem a convergência, foi aplicado o modelo *copybest* geralmente utilizado no GA. Este processo simplesmente copia os melhores cromossomos de uma geração para outra. No caso da adaptação ao PSO, não haveria cálculo nem atualização para as melhores partículas. Isto não surtiu o efeito esperado, mas abriu caminho para começar a dar um tratamento diferenciado para as melhores partículas. Então foram estabelecidas constantes c_1 e c_2 com valores de 1,7 para as 10 melhores partículas, menores que os c_1 e c_2 de 1,9 usadas no restante do enxame. Ou seja, aliviamos a pressão de busca para as melhores e aí sim tivemos o efeito esperado: a otimização forneceu resultados razoáveis, mostrando a viabilidade do processo (MENESES e SCHIRRU, 2005).

Seja *P* o número total de partículas.
Seja *x(i,j)* a posição $\mathbf{x_{ij}}^{t}$ da i-ésima partícula.
Seja *vetor_aux(i,j)* um vetor auxiliar $\mathbf{u_{ij}}^{t}$ para a i-ésima partícula, representando uma solução candidata para o TSP.
Seja *fitness(i)* a função objetivo $f(\mathbf{u_{ij}}^{t})$.
Seja *v(i,j)* a velocidade $\mathbf{v_{ij}}^{t}$ da partícula i.
Seja *gbest(j)* a melhor posição global $\mathbf{gbest_j}$.
Seja *bestfitnessglobal* a melhor fitness, encontrada a partir de *gbest(j)*.
Seja *pbest(i,j)* a melhor posição $\mathbf{pbest_{ij}}$ da i-ésima partícula.
Seja *bestfitnesscadapart(i)* a melhor fitness da partícula i, encontrada a partir de *pbest(i,j)*.

Passo 1 (Inicialização):

1.1 Inicializar *x(i,j)* randomicamente no intervalo (0,48).
1.2 Inicializar *vetor_aux(i,j)* randomicamente.
1.3 Inicializar *v(i,j)* randomicamente.
1.4 Inicializar *pbest(i,j)*como uma cópia de *x(i,j)*.
1.5 Inicializar *gbest(j)* com *x(i,j)* da partícula i de menor fitness.

Passo 2: Repetir até que algum critério seja satisfeito.

2.1 Reinicializar aleatoriamente uma partícula escolhida ao acaso (sub-rotina *mata_abelhas*).
2.2 Definir as 10 melhores partículas e as 5 piores.
2.3 Para cada partícula i
(a) se i é uma das 5 piores, então ela assume a configuração da melhor partícula.
(b) calcular *v(i,j)* de acordo com a eq. (6) com constantes diferenciadas para as 10 melhores partículas.
(c) Atualizar *x(i,j)* de acordo com a equação (7).
(d) Ordenar *vetor_aux(i,j)* usando RK, com as chaves fornecidas por *x(i,j)*.
(e) Avaliar *fitness(i)* em função de *vetor_aux(i,j)*.
(f) Se *fitness(i)* ≤ *bestfitnesscadapart(i)* então *pbest(i,j)* ← *x(i,j)*;
(g) Se *fitness(i)*≤*bestfitnessglobal* então *gbest(j)* ← *x(i,j)*

Fig. 4.6: Algoritmo do PSO para otimização do Rykel 48, para enxames reduzidos.

Houve também uma modificação na equação (7), na tentativa de auxiliar o processo de convergência: em vez de o cálculo de uma nova posição ser feito com base na posição anterior, seria efetuado com base na sua melhor posição até o momento (estratégia Melhor Posição). Deve ser lembrado que a posição nada mais é do que um conjunto de chaves utilizadas na decodificação. Ou seja, o processo seria feito com base na codificação que forneceu o melhor vetor auxiliar, ou seja, de menor *fitness*. Entretanto, tal procedimento não proporcionou melhora na convergência nos experimentos.

Uma outra tentativa de chegar a resultados regulares foi a inclusão de um operador chamado Mudança de Configuração. Este operador seleciona um certo número de partículas, aquelas que obtiveram pior avaliação de fitness. Depois de selecionadas, elas alteram sua configuração imitando configuração da melhor partícula. Chamamos essas partículas de "invejosas" pois buscam adquirir aspectos da melhor sem, no entanto, terem informações de que caminho foi percorrido pela partícula para que ela chegasse naquele ponto. No próximo capítulo serão mostrados gráficos para sua análise. Também no algoritmo para otimização do Rykel, utilizamos a já mencionada sub-rotina *mata_abelhas*, que elimina uma partícula e a reinicializa.

Vale destacar que, para o Rykel 48, as melhores posições individuais $\mathbf{pbest_{ij}}$ e global $\mathbf{gbest_j}$ são obtidas a partir dos vetores de posições $\mathbf{x_{ij}}$, a exemplo do algoritmo básico do PSO, em vez de serem obtidas com base nos vetores auxiliares $\mathbf{u_{ij}}$ destinados ao cálculos das avaliações, como para o Oliver 30. Desta forma, também são utilizadas para o Rykel 48 as equações (6) e (7), com $w \in \{w_1, w_2\}$, $c_1 \in \{c_{11}, c_{12}\}$ e $c_2 \in \{c_{21}, c_{22}\}$. Se a partícula em questão pertencer ao conjunto das 10 melhores partículas em uma geração t então serão utilizadas as constantes w_1, c_{11} e c_{21} para o cálculo da geração t+1; em caso contrário serão utilizados w_2, c_{12} e c_{22}.

Estas formulações permitiram chegar a bons resultados para o TSP Rykel 48, embora um pouco diferentes daquelas elaboradas para a otimização do Oliver 30. MEDEIROS (2005) mostra resultados obtidos para um enxame contendo um número maior de partículas. Com base nisto, também realizamos testes para enxames com 500 e 1000 partículas, em vez de utilizarmos em torno de 100, da forma relatada no subitem seguinte.

4.3.2. Otimização do Rykel 48 com Enxames Expandidos

Nesta etapa do trabalho, na qual foram usados enxames de 500 e 1000 partículas (que convencionamos chamar *enxames expandidos*), vale ressaltar que, além de ter havido redução considerável no número de iterações necessárias até serem obtidos resultados satisfatórios, foi possível eliminar os métodos que embora apoiando e melhorando o processo não faziam parte da proposta inicial do PSO. Assim, esta alteração mostrou abrir caminhos para a unificação entre os métodos apresentados para os dois problemas, podendo-se chegar futuramente a um único modelo para PSO para o problema combinatório, independentemente do número de cidades ou do tipo de decodificação utilizado para os vetores das equações.

Seja *P* o número total de partículas.
Seja *x(i,j)* a posição $\mathbf{x_{ij}^t}$ da i-ésima partícula.
Seja *vetor_aux(i,j)* um vetor auxiliar $\mathbf{u_{ij}^t}$ para a i-ésima partícula, representando uma solução candidata para o TSP.
Seja *fitness(i)* a função objetivo f($\mathbf{u_{ij}^t}$).
Seja *v(i,j)* a velocidade $\mathbf{v_{ij}^t}$ da partícula i.
Seja *gbest(j)* a melhor posição global $\mathbf{gbest_j}$.
Seja *bestfitnessglobal* a melhor fitness, encontrada a partir de *gbest(j)*.
Seja *pbest(i,j)* a melhor posição $\mathbf{pbest_{ij}}$ da i-ésima partícula.
Seja *bestfitnesscadapart(i)* a melhor fitness da partícula i, encontrada a partir de *pbest(i,j)*.

Passo 1 (Inicialização):
- **1.1** Inicializar *x(i,j)* randomicamente no intervalo (0,48).
- **1.2** Inicializar *vetor_aux(i,j)* randomicamente.
- **1.3** Inicializar *v(i,j)* randomicamente.
- **1.4** Inicializar *pbest(i,j)*como uma cópia de *x(i,j)*.
- **1.5** Inicializar *gbest(j)* com *x(i,j)* da partícula i de menor fitness.

Passo 2: Repetir até que algum critério seja satisfeito.
- **2.1** Para cada partícula i
  - (a) calcular *v(i,j)* de acordo com a eq. (6).
  - (b) Atualizar *x(i,j)* de acordo com a equação (7).
  - (c) Ordenar *vetor_aux(i,j)* usando RK, com as chaves fornecidas por x*(i,j)*.
  - (d) Avaliar *fitness(i)* em função de *vetor_aux(i,j)*.
  - (e) Se *fitness(i)* ≤ *bestfitnesscadapart(i)* então *pbest(i,j)* ← *x(i,j)*;
  - (f) Se *fitness(i)*≤*bestfitnessglobal* então *gbest(j)* ← *x(i,j)*

Fig. 4.7: Algoritmo do PSO para otimização do Rykel 48 usando enxames expandidos.

As equações utilizadas para este caso também são as (6) e (7) e o pseudocódigo referente a este método é mostrado na figura 4.7.

A apresentação dos resultados tanto para o Oliver 30 quanto para o Rykel 48, comparações, análises e discussões serão apresentadas no próximo capítulo.

Capítulo 5

Resultados dos Experimentos e Análises

Este capítulo mostra o resultado dos experimentos realizados a partir da implementação dos pseudocódigos apresentados no capítulo anterior, destinados à otimização dos TSPs Oliver 30 e Rykel 48, com as respectivas análises.

Para cada problema, são mostrados dois grupos principais de resultados: um para o que convencionamos chamar *enxames reduzidos* (em torno de 100 partículas) e outro para *enxames expandidos*, com um número maior de partículas (usando 500 e 1000 partículas). As tabelas e gráficos apresentados para enxames reduzidos mostram resultados que foram sendo aprimorados ao longo do processo, de acordo com o andamento das investigações realizadas, chegando a bons mínimos locais, comparáveis aos obtidos por outros métodos, sem, entretanto, obter uma regularidade que garantisse a robustez do método. Já os enxames expandidos permitiram que diversos experimentos fornecessem resultados mais consistentes, chegando a mínimos globais ou próximo deles com uma regularidade maior. Portanto, para enxames reduzidos mostraremos os melhores resultados, em experimentos específicos com o intuito de ilustrarmos os resultados obtidos ao longo das etapas de desenvolvimento da técnica. Para enxames expandidos serão mostrados os melhores resultados seguidos das médias dos resultados de diversos experimentos realizados. Para o Rykel 48, eliminamos os procedimentos diferenciados para retornarmos ao máximo aos traços do modelo original de PSO mostrado no item 2.2, com o pseudocódigo da figura 4.7. Outros gráficos comparativos também serão mostrados.

Os códigos foram implementados em FORTRAN estruturado no pacote Digital Visual Fortran.

5.1. Resultados para o Oliver 30 e o Rykel 48

5.1.1. Tabelas e gráficos para Oliver 30 usando enxames reduzidos

As tabelas abaixo mostram os resultados obtidos com duas sementes distintas, com as constantes w = 0,06; $c_1 = c_2 = 0,1$, que, embora atípicos em relação aos mencionados na literatura, foram os que forneceram melhores resultados nos vários testes realizados conforme método descrito no item 4.2 do capítulo anterior, com enxames contendo de 60 a 120 partículas, conforme descrito nas tabelas seguintes.

Número de Partículas	50 iterações		100 iterações		150 iterações		200 iterações	
	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
60	746,77	532,93	628,02	517,91	620,75	517,19	630,70	507,87
80	661,45	479,47	529,63	424,77	497,63	423,94	474,22	423,73*
100	506,62	428,89	467,77	424,77	461,87	424,56	461,97	424,56
120	526,71	427,20	470,05	423,93	454,98	423,73*	457,58	423,73*

* Mínimo global (DORIGO e GAMBARDELLA, 1996).

Tabela 5.1. Grupo de resultados para enxames reduzidos e, experimento com as constantes w = 0,06 e $c_1 = c_2 = 0,1$.

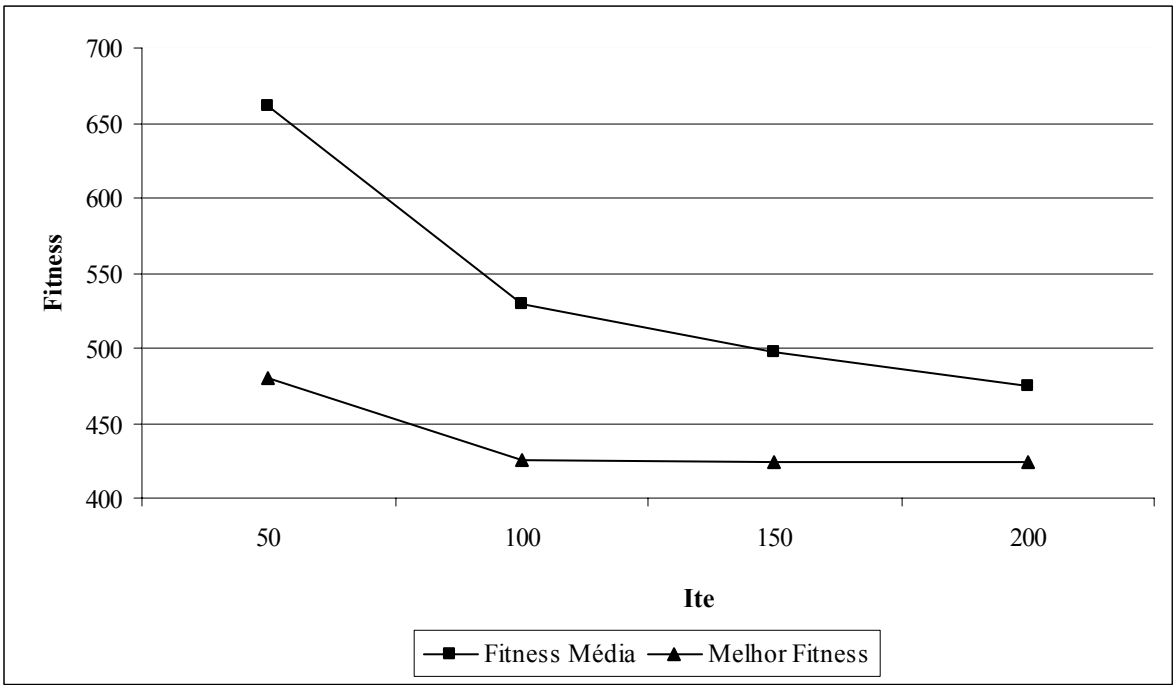


Gráfico 5.1: Resultados da otimização do Oliver 30, com um enxame de 80 partículas do experimento mostrado na tabela 5.1, com obtenção do mínimo global 423,73.

Número de Partículas	50 iterações		100 iterações		150 iterações		200 iterações	
	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
60	745,26	473,89	501,05	424,66	483,68	424,66	506,87	424,11
80	639,45	507,17	551,51	456,44	525,28	456,23	516,01	456,23
100	508,64	424,11	464,62	423,94	467,74	423,73*	467,78	423,73*
120	514,12	425,50	459,57	423,94	452,41	423,94	458,77	423,93

* Mínimo global (DORIGO e GAMBARDELLA, 1996).

Tabela 5.2. Grupo de resultados para enxames reduzidos, em outro experimento com as mesmas constantes do grupo anterior.

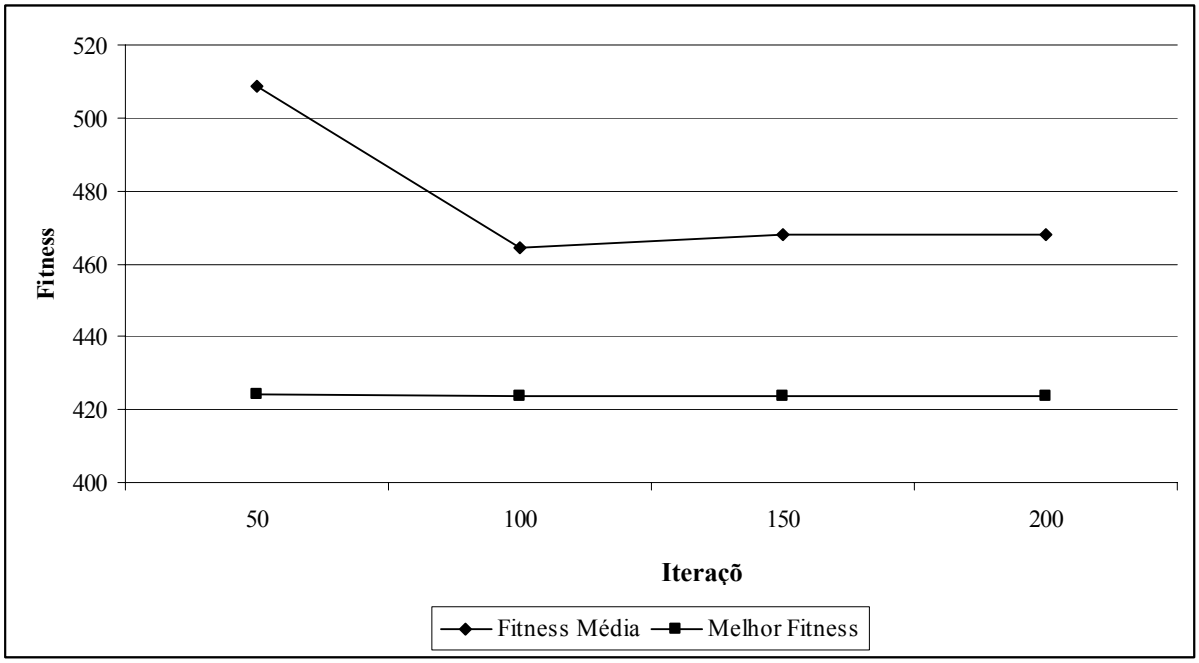


Gráfico 5.2: Resultados da otimização do Oliver 30, com um enxame de 100 partículas, do experimento mostrado na tabela 5.2, com obtenção do mínimo global 423,73.

5.1.2. Tabelas e gráficos para Oliver 30 usando enxame expandido

O algoritmo e as constantes utilizados para a otimização com 500 e 1000 partículas são os mesmo utilizados nos experimentos do item 5.1.1.

5.1.2.1. Oliver 30 com enxame de 500 partículas

A tabela 5.3 mostra o melhor grupo de resultados para enxames de 500 partículas.

Iterações	Melhor fitness	Fitness média do enxame
20	531,59	988,81
40	425,82	604,92
60	423,95	515,14
80	423,73*	476,91
100	423,73*	464,96

* Mínimo global (DORIGO e GAMBARDELLA, 1996).

Tabela 5.3. Melhor grupo de resultados para enxames de 500 partículas.

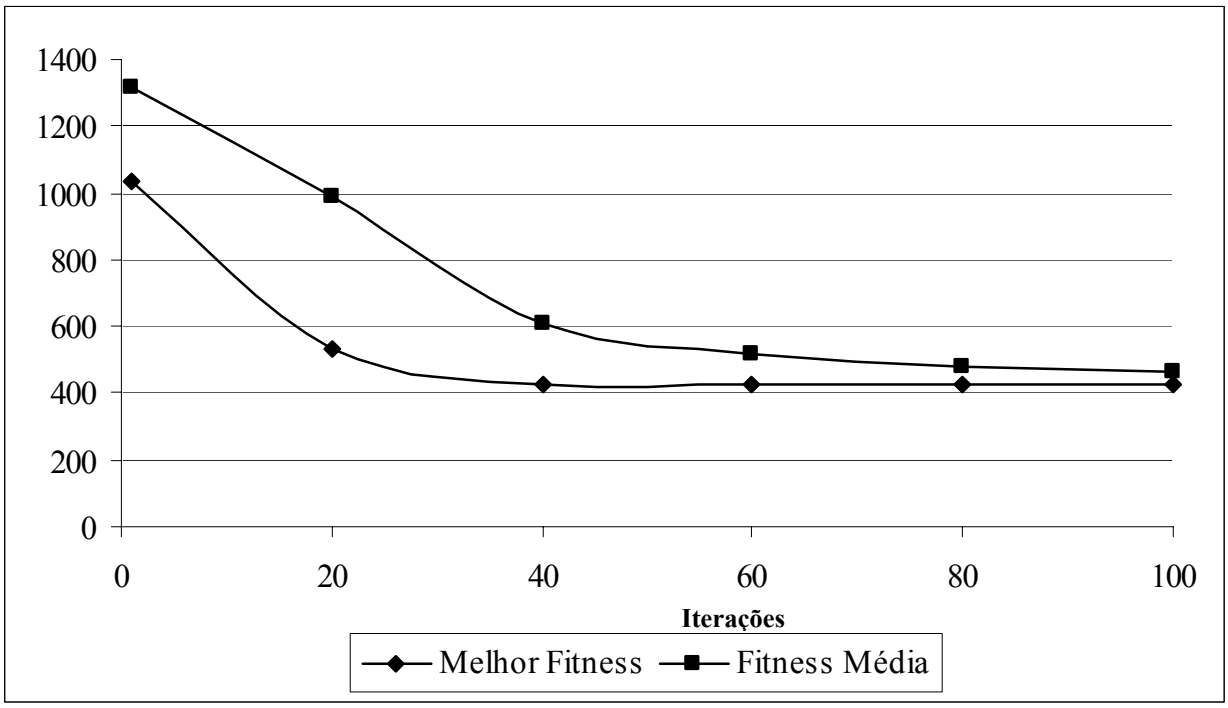


Gráfico 5.3: Melhor resultado da otimização do Oliver 30, com um enxame de 500 partículas, com obtenção do mínimo global 423,73.

Este experimento com enxames de 500 partículas foi realizado com 10 sementes distintas utilizando-se as mesmas constantes, a fim de se verificar a robustez do método. Abaixo está a tabela com os valores médios de fitness a cada iteração indicada e a média das fitness médias de cada enxame.

Iterações	Média das melhores fitness	Média das fitness médias dos enxames
20	564,41	884,90
40	453,60	614,53
60	443,33	539,60
80	441,63	512,05
100	441,57	502,22

Tabela 5.4. Resultados médios para 10 experimentos realizados, todos com as mesmas constantes, sementes distintas e enxames de 500 partículas para o Oliver 30.

O mínimo global foi atingido em menos de 100 gerações em oito dos dez experimentos realizados. Uma das sementes proporcionou um resultado muito alto, o que elevou a média das melhores fitness para 441,57, como mostra a última linha da tabela. O gráfico correspondente encontra-se abaixo.

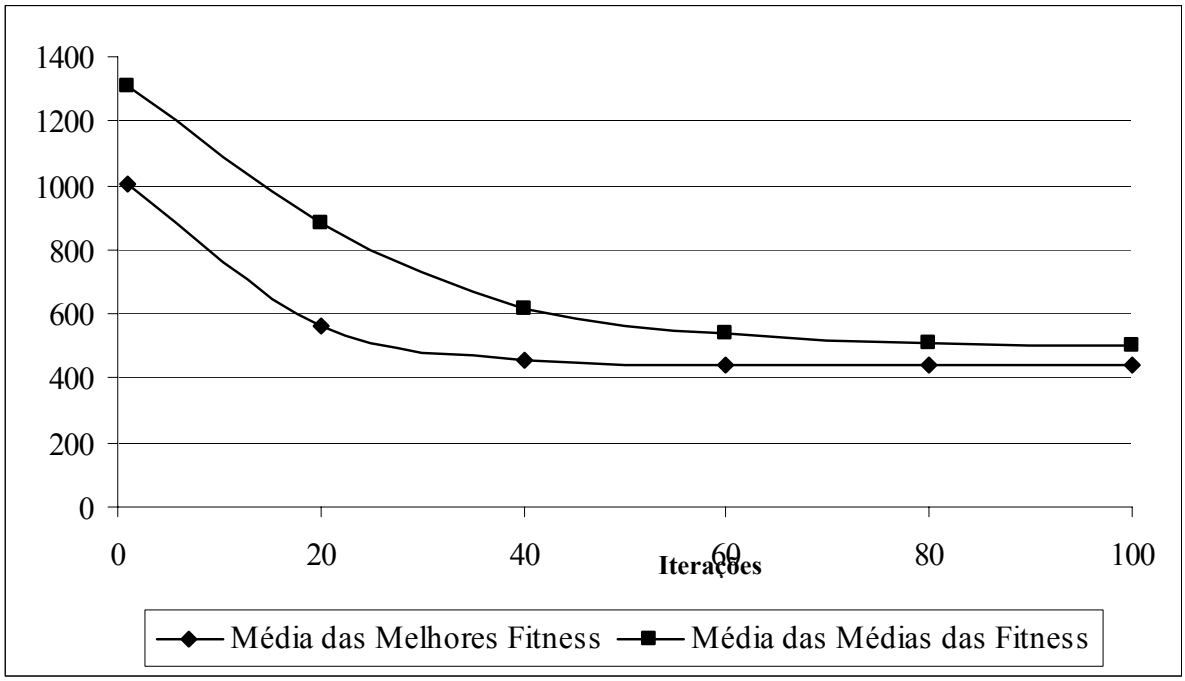


Gráfico 5.4: Resultados médios da otimização do Oliver 30, para 10 experimentos com enxames de 500 partículas.

5.1.2.2. Oliver 30 com enxame de 1000 partículas

A tabela abaixo mostra a evolução do experimento de melhor desempenho para enxames de 1000 partículas com as constantes $w = 0,06$ e $c_1 = c_2 = 0,1$.

Iterações	Melhor fitness	Fitness média do enxame
20	460,42	666,83
40	423,95	508,83
60	423,73*	463,03
80	423,73*	448,68
100	423,73*	446,22

* Mínimo global (DORIGO e GAMBARDELLA, 1996).

Tabela 5.5. Melhor grupo de resultados para população de 1000 partículas.

O gráfico correspondente é mostrado abaixo:

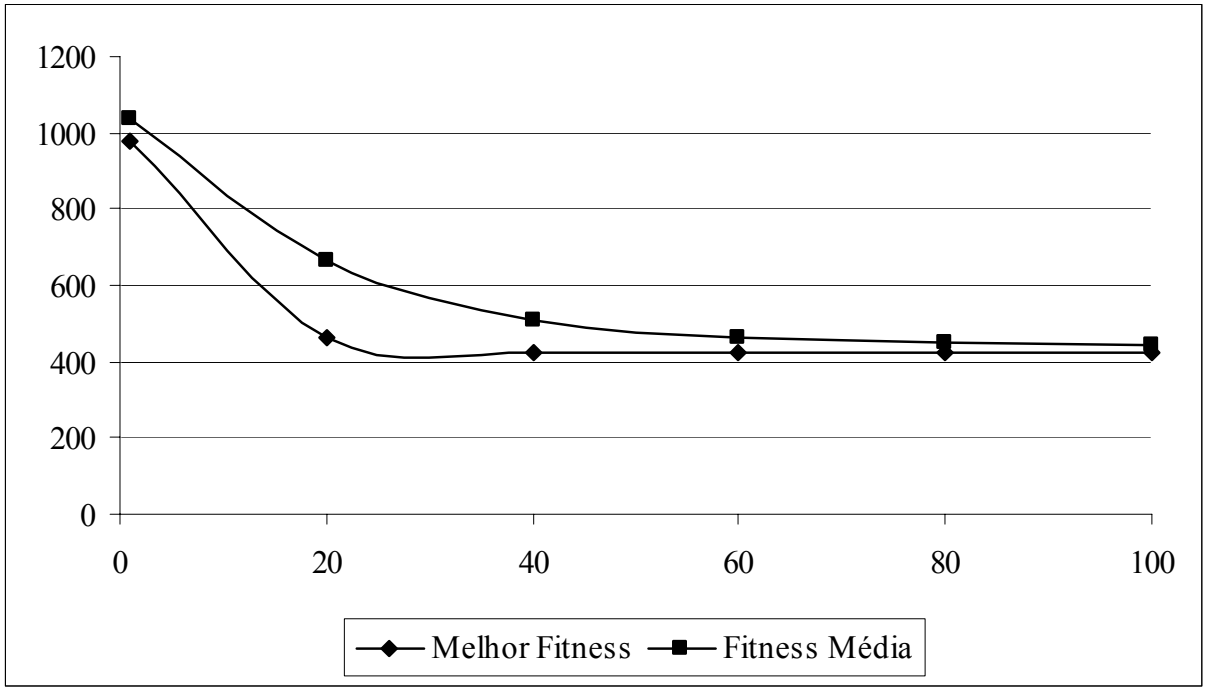


Gráfico 5.5: Melhor resultado obtido na otimização do Oliver 30, em experimentos com enxames de 1000 partículas.

Este experimento também foi realizado com 10 sementes distintas utilizando-se as mesmas constantes. Abaixo encontra-se a tabela com os valores médios de fitness a cada geração indicada e a média das fitness médias de cada enxame:

Iterações	Melhor fitness	Fitness média do enxame
20	512,91	826,12
40	431,255	554,48
60	424,01	481,61
80	423,84	455,09
100	423,76	447,57

Tabela 5.6. Resultados médios para otimização do Oliver 30, a partir de 10 enxames de 1000 partículas.

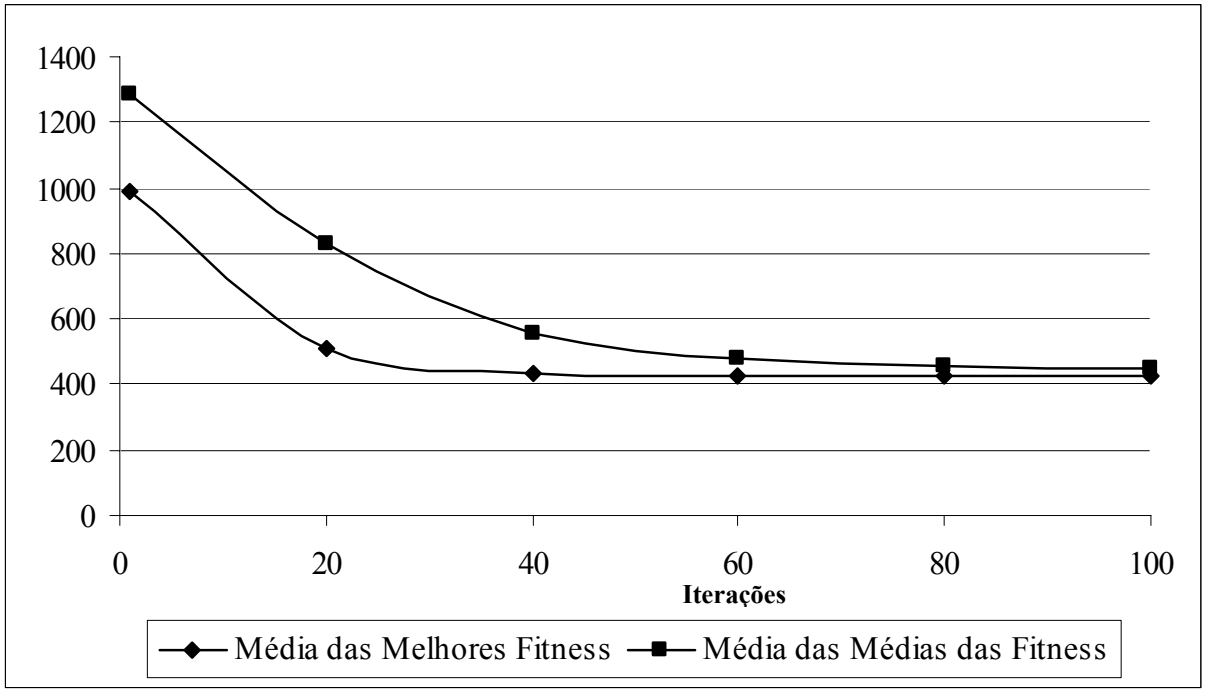


Gráfico 5.6: Resultados médios da otimização do Oliver 30, para 10 experimentos com enxames de 1000 partículas.

5.1.3. Tabelas e gráficos para o Rykel 48 utilizando enxames reduzidos

5.1.3.1. Primeiro importante resultado com enxames reduzidos

Um importante resultado foi obtido com enxame reduzido de 120 partículas como é mostrado na tabela 5.7, mostrando a viabilidade do método. O valor de mínimo de 15.412 (MENESES e SCHIRRU, 2005) é considerável para um problema da magnitude do Rykel 48, cujo mínimo é 14.422 (MACHADO, 1999). Entretanto, não foi possível obter uma regularidade entre os valores mínimos ao longo de 50.000 gerações como pode ser visto na tabela 5.8. A partir disto buscou-se aprimorar o método em busca de uma regularidade maior de mínimos encontrados.

Os dados das tabelas 5.7 e 5.8 foram obtidos com o algoritmo da figura 4.6, apenas sem o item (a) da linha 2.3, que é o operador de mudança de configuração, para o qual temos os resultados do item 5.1.3.2. Foram utilizadas as constantes w = 0,75; $c_{11} = c_{21} = 1,7$ e $c_{12} = c_{22} = 1,9$.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50000 iterações	
	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness Média	Melhor fitness	Fitness média	Melhor fitness
100	40352	18445	39382	17531	39319	17531	39015	17352
120	45090	23542	41369	16680	40802	15462	40410	15412*
140	50916	25736	50368	24979	49667	23908	49722	23758
160	41243	21122	40889	19809	40678	19607	40245	19404

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.7: Primeiro grupo de resultados importantes obtidos para o Rykel 48 em uma série de experimentos com enxames reduzidos

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness média	Melhor fitness	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	45408	21950	45031	20525	45168	20338	44947	20276
120	42378	19657	41458	19438	42279	19371	41895	18896*
130	51319	23356	50452	21321	50609	20017	50471	20017
140	46139	24429	46152	23100	45419	23079	45091	23015

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.8: Outro grupo de resultados em outra série de experimentos, com outras sementes. Nota-se que ainda não há homogeneidade entre os resultados, comparando-se aos mínimos locais mostrados na tabela 5.7.

5.1.3.2. Resultados para o Rykel 48 usando o operador Mudança de Configuração.

Os resultados das tabelas abaixo foram obtidos usando as equações (21) e (22), com a inicialização das posições no intervalo (0, 48), a subrotina *mata_abelhas* e o operador de Mudança de Configuração (item 4.3). Usando estas estratégias de maneira associada, obtivemos uma regularidade e consistência maior nos resultados, mesmo usando enxames reduzidos.

Abaixo, no item (A), mostramos alguns grupos de resultados com variação no número de partículas invejosas. O item (B) traz um número fixo de cinco partículas que copiam a configuração da melhor partícula da iteração. Os gráficos dos desempenhos dos três melhores enxames são exibidos no item (C). No item (D) mostramos os resultados para o melhor experimento obtido anteriormente realizando um número bem maior de iterações. O item (E) traz um gráfico comparativo entre dois experimentos com diferentes sementes, mas com as mesmas condições gerais: o mesmo número de partículas e as mesmas constantes, um deles trabalhando com e o outro sem o operador Mudança de Configuração.

(A) Resultados obtidos com variação no número de partículas invejosas de acordo com o tamanho da população.

Número de total de partículas no enxame	Número de partículas invejosas
110	0
120	5
130	15
140	25

Tabela 5.9: Número de partículas *invejosas* em cada enxame para os resultados do item 5.1.3.2(A).

Neste item são apresentados os resultados de testes com 4 tipos de população, cada uma com um número diferente de partículas invejosas, conforme mostra a tabela 5.9.

Foram escolhidos 3 grupos de constantes, sendo feitos dois experimentos para cada um deles.

(I) Grupo A: constantes w = 0,7; $c_{11} = c_{21} = 1,65$ e $c_{12} = c_{22} = 1,9$.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness média	Melhor fitness	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	23329	19798	23111	19397	25076	19392	24905	19392
120	30034	17997	27601	16631	27411	16096	26816	16096*
130	33667	18877	33617	18085	33486	17953	33306	17953
140	36041	18955	33376	17429	31636	17059	30920	16944

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.10: Resultados para o Rykel 48 com operador de mudança de configuração variando de acordo com o tamanho da população: experimento 1 com o grupo A de constantes.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness Média	Melhor fitness	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	32210	24356	32077	24196	32386	24196	31945	24196
120	31986	19160	32813	18925	32826	18871	32697	18575
130	29312	16772	28695	15918	28253	15918	28596	15757*
140	32424	18336	32146	18099	31295	18042	31319	17812

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.11: Resultados para o Rykel 48 com operador de mudança de configuração variando de acordo com o tamanho da população: experimento 2 com o grupo A de constantes.

(II) Grupo B: constantes w = 0,6; $c_{11} = c_{21} = 1,6$ e $c_{12} = c_{22} = 1,85$.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness Média	Melhor fitness	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	26613	19642	24379	19627	24389	19132	24389	19093
120	25936	16952	26569	16626	26670	16626	26379	16626*
130	30725	18429	29275	18347	29378	18253	29350	17255
140	31827	19689	32386	19565	32473	19406	32770	19327

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.12: Resultados para o Rykel 48 com operador de mudança de configuração variando de acordo com o tamanho da população: experimento 1 com o grupo B de constantes.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	25022	18694	26759	18497	24973	18398	24700	18398
120	27268	18489	27626	17656	27381	17279	27834	17065*
130	26770	18414	26396	18414	27165	18085	27058	18021
140	27860	19042	27595	18611	26946	18611	27051	18611

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.13: Resultados para o Rykel 48 com operador de mudança de configuração variando de acordo com o tamanho da população: experimento 2 com o grupo B de constantes.

(III) Grupo C: constantes w = 0,7; $c_{11} = c_{21} = 1,65$ e $c_{12} = c_{22} = 1,88$.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	33208	24479	32031	23753	31645	23706	31631	23706
120	31125	18720	31515	18299	32280	18141	31659	18141
130	35701	21947	36240	20483	35576	19240	36194	19076
140	26980	16399	26692	15990	26931	15990	27740	15885*

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.14: Resultados para o Rykel 48 com operador de mudança de configuração variando de acordo com o tamanho da população. Experimento 1 com o grupo C de constantes.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	33423	25114	32779	24704	32486	24704	33881	24643
120	26745	16988	25904	16231	25491	16194	25337	16194*
130	26646	17021	26966	16998	26845	16920	27833	16443
140	34543	21385	35051	20952	34563	20206	33321	19156

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.15: Resultados para o Rykel 48 com operador de mudança de configuração variando de acordo com o tamanho da população. Experimento 2 com o grupo C de constantes.

(B) Resultados para um número fixo de partículas invejosas.

Neste caso todos os enxames utilizados para testes possuíam 5 partículas invejosas. Foram utilizados os mesmos grupos de constantes, sendo realizados dois experimentos para cada um deles.

(I) Grupo A: constantes w = 0,7; $c_{11} = c_{21} = 1,65$ e $c_{12} = c_{22} = 1,9$.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	26503	16435	25208	16060	24540	16060	25191	15620*
120	30034	17997	27601	16631	27412	16096	26817	16096
130	35368	21190	35929	20820	36436	19808	36338	19808
140	32977	19219	30976	18495	31438	18254	31276	18124

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.16: Resultados para o Rykel 48 com operador de mudança de configuração, com um número fixo de 5 partículas invejosas. Experimento 1 com o grupo A de constantes.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	39329	19556	40664	18460	39352	18460	39616	18053
120	31986	19160	32813	18925	32826	18871	32697	18575
130	34443	19208	34878	18252	34815	17606	35539	17158*
140	36762	19498	36255	19284	36517	18954	36246	18041

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.17: Resultados para o Rykel 48 com operador de mudança de configuração, com um número fixo de 5 partículas invejosas. Experimento 2 com o grupo A de constantes.

(II) Grupo B: constantes w = 0,6; $c_{11} = c_{21} = 1,6$ e $c_{12} = c_{22} = 1,85$.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	30787	18964	30710	18811	29727	18811	29960	18811
120	25936	16952	26569	16626	26670	16626	26379	16626*
130	25881	18671	25265	18297	25454	18299	24794	18299
140	24850	18869	25835	17527	24682	17007	25159	17007

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.18: Resultados para o Rykel 48 com operador de mudança de configuração, com um número fixo de 5 partículas invejosas: Experimento 1 com o grupo B de constantes.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	22746	17177	23068	16896	23696	16548	23465	16391*
120	27268	18489	27626	17656	27381	17279	27834	17065
130	31895	18031	31920	17741	32412	17663	31550	17427
140	28212	19945	28118	19874	28652	19765	30562	19285

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.19: Resultados para o Rykel 48 com operador de mudança de configuração, com um número fixo de 5 partículas invejosas. Experimento 2 com o grupo B de constantes.

(III) Grupo C: constantes w = 0,7; $c_{11} = c_{21} = 1,65$ e $c_{12} = c_{22} = 1,88$.

Número de Partículas	12.500 iterações	25.000 iterações	37.500 iterações	50.000 iterações

Partículas	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	27357	15918	27270	15723	27039	15723	27158	15381*
120	31125	18720	31515	18299	32280	18141	31659	18141
130	37558	21831	34791	20676	34506	20651	34810	20113
140	27890	17254	28004	16886	27614	16886	27692	16787

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.20: Resultados para o Rykel 48 com operador de mudança de configuração, com um número fixo de 5 partículas invejosas. Experimento 1 com o grupo C de constantes.

Número de Partículas	12.500 iterações		25.000 iterações		37.500 iterações		50.000 iterações	
	Fitness Média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness	Fitness média	Melhor fitness
110	24118	16629	22931	16364	22873	16364	24021	16245
120	26745	16988	25904	16231	25491	16194	25337	16194*
130	31973	19696	31988	18918	31915	18918	31715	18918
140	41268	23330	41061	22302	38582	21501	38438	21053

* Melhor resultado para o experimento em 50.000 gerações.

Tabela 5.21: Resultados para o Rykel 48 com operador de mudança de configuração, com um número fixo de 5 partículas invejosas. Experimento 2 com o grupo C de constantes.

(C) Gráficos para os três melhores experimentos.

(I) semente 123456789; 110 partículas; grupo C de constantes w = 0,7; $c_{11} = c_{21} =$ 1,65 e $c_{12} = c_{22} = 1,88$; com 5 partículas invejosas (tabela 5.20).

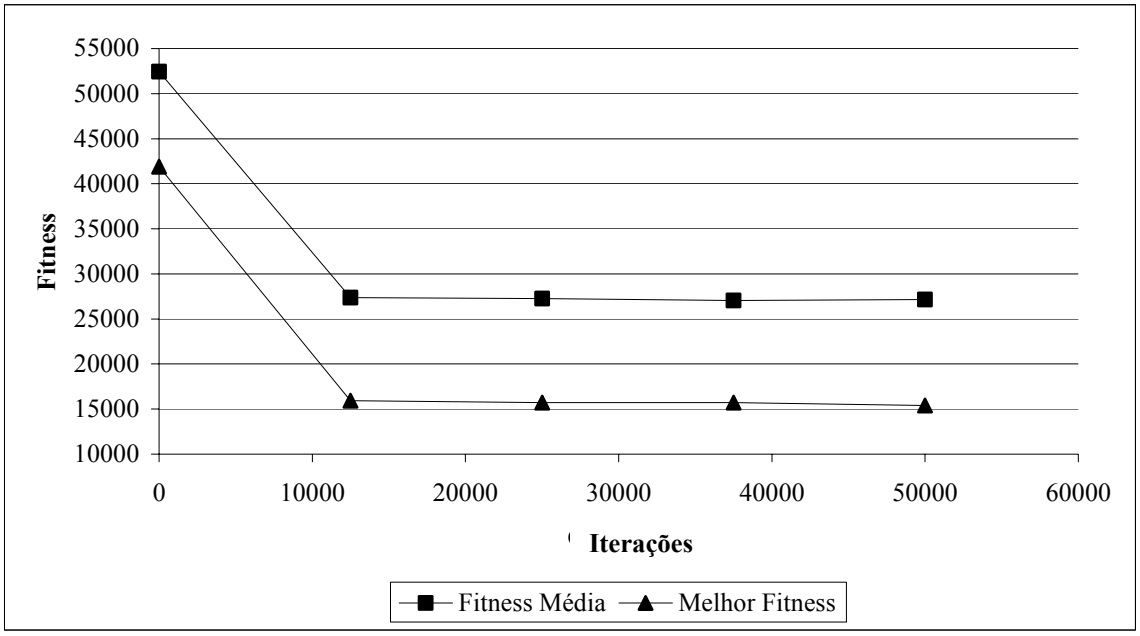


Gráfico 5.7: Resultados da otimização do Rykel 48, com um enxame de 110 partículas, semente iseed=123456789 e grupo C de constantes, com obtenção do

(II) semente 123456789; 110 partículas; grupo A de constantes w = 0,7; $c_{11} = c_{21} =$ 1,65 e $c_{12} = c_{22} = 1,9$; com 5 partículas invejosas (tabela 5.16).

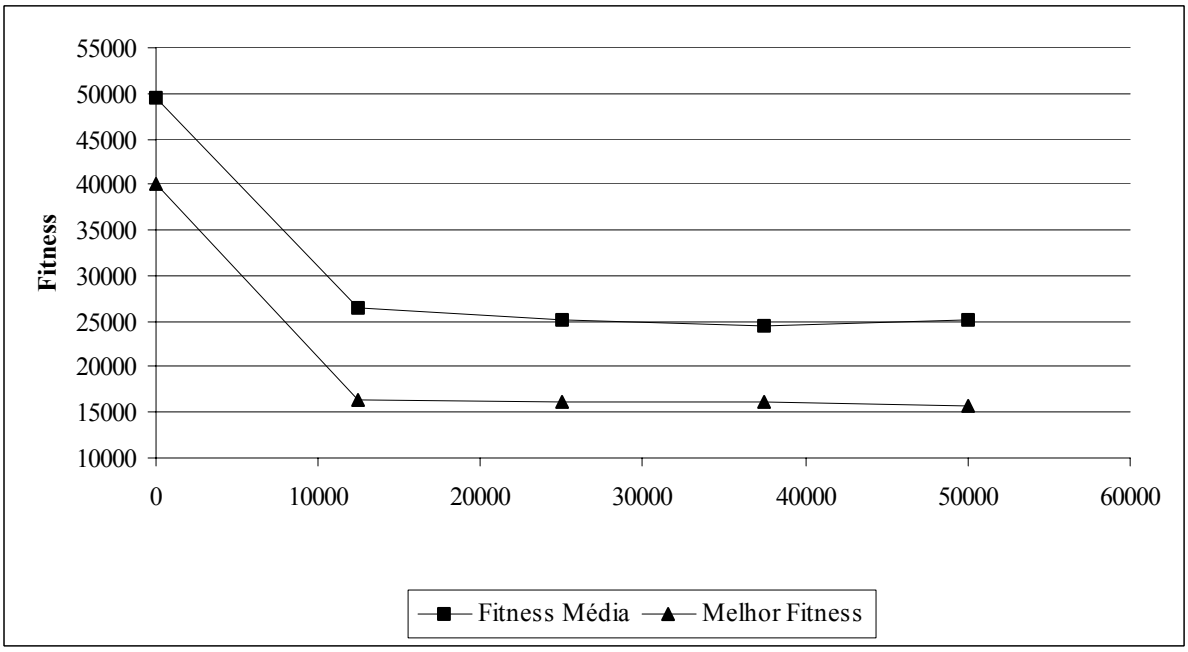


Gráfico 5.8: Resultados da otimização do Rykel 48, com um enxame de 110 partículas, semente iseed=123456789 e grupo A de constantes, com obtenção do

(III) semente 52435; 130 partículas; grupo A de constantes w = 0,7; $c_{11} = c_{21} =$ 1,65 e $c_{12} = c_{22} = 1,9$; com 15 partículas invejosas (tabela 5.11).

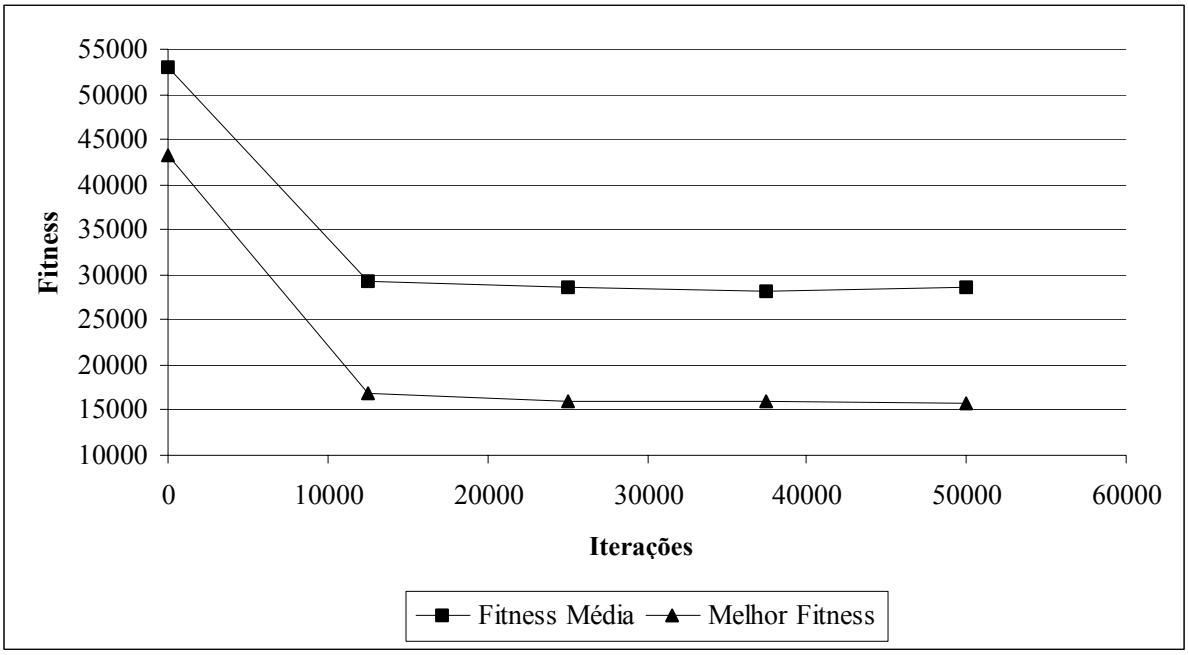


Gráfico 5.9: Resultados da otimização do Rykel 48, com um enxame de 110 partículas e semente iseed=123456789 e grupo C de constantes, com obtenção do mínimo local

(D) Resultados para um enxame de 120 partículas, para 125.000 gerações.

Os resultados deste item mostram o desempenho global de apenas um enxame ao longo de 125.000 gerações. Até a geração 50.000, inclusive, são os mesmos constantes na tabela 5.21 acima, para 120 partículas. Foram utilizados a semente 52435 o grupo C de constantes w = 0,7; $c_{11} = c_{21} = 1,65$ e $c_{12} = c_{22} = 1,88$.

Iteração	12500	25000	37500	50000	62500	75000	87500	100000	112500	125000
Melhor Fitness	**16988**	**16231**	**16194**	**16194**	**16194**	**15662**	**14929**	**14720**	**14706**	**14686**
Partícula	78	108	104	104	104	99	79	99	99	30

Tabela 5.22: Resultados para 125000 gerações, para um enxame com 110 partículas.

Este resultado foi o melhor registrado durante os testes, devido ao prolongado número de iterações. Seu processamento não demorou mais que 10 minutos em um processador Intel Celeron 1,8GHz. A melhor seqüência de cidades obtida foi: 7 - 6 - 19 - 27 - 17 - 43 - 37 - 30 - 28 - 36 - 44 - 31 - 38 - 40 - 9 - 1 - 8 - 22 - 16 - 3 - 41 - 34 - 29 -

2 - 42 - 26 - 4 - 35 - 45 - 10 - 24 - 32 - 39 - 48 - 5 - 25 - 14 - 23 - 11 - 13 - 21 - 47 - 20 - 33 - 12 - 15 - 46 - 18.

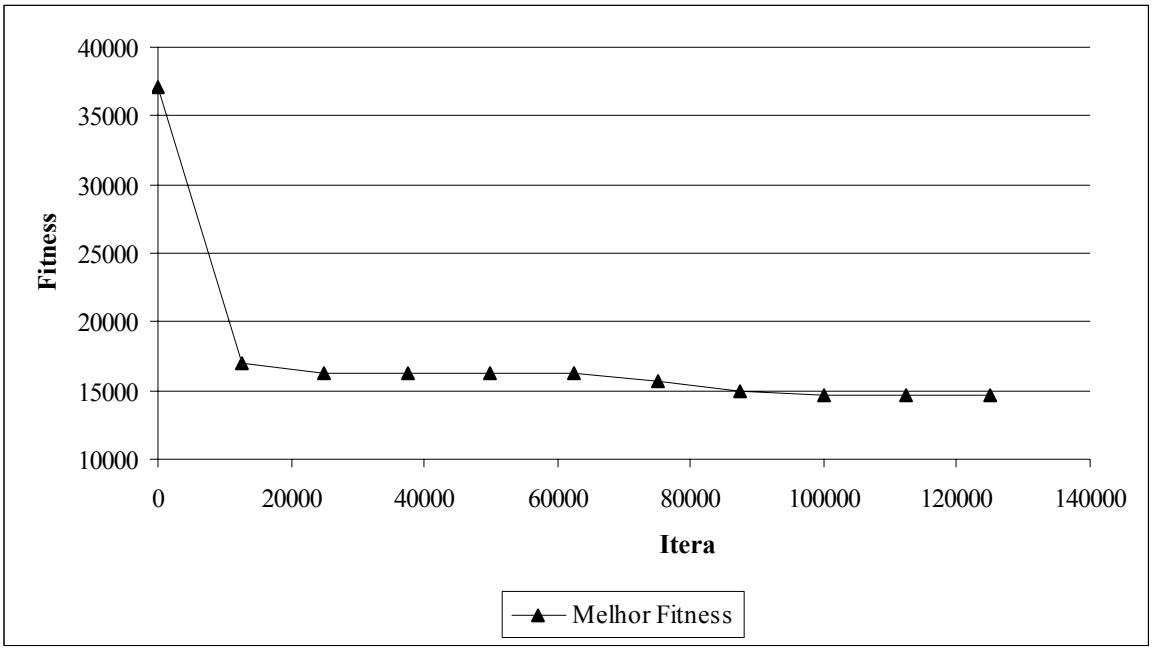


Gráfico 5.10: Melhor desempenho obtido na otimização do Rykel 48. Resultado para um enxame de 120 partículas, ao longo de 125.000 gerações.

(E) Gráfico comparativo entre os resultados obtidos com e sem o operador de mudança de configuração (para um enxame de 110 partículas).

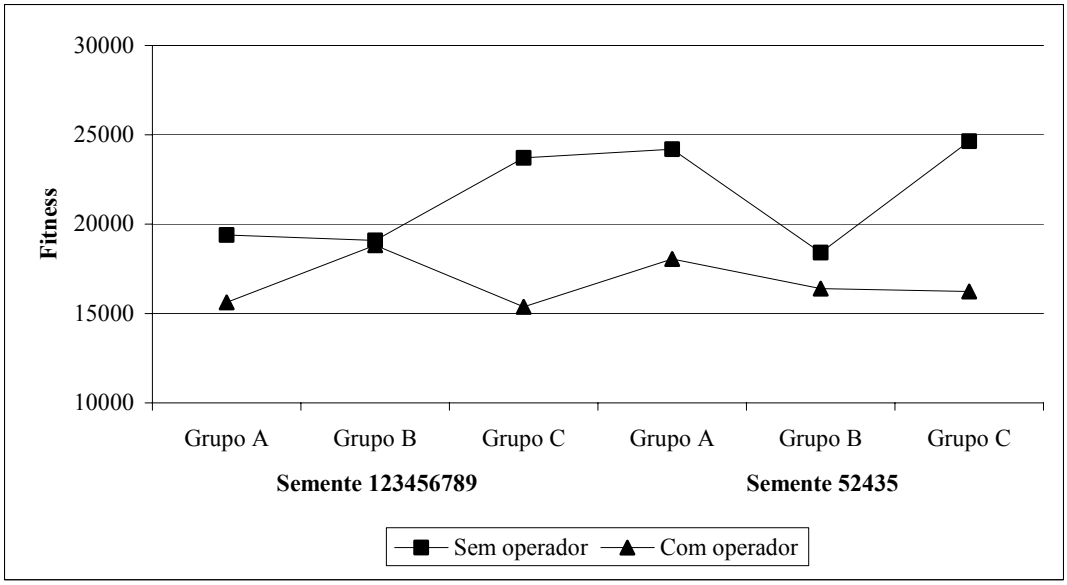


Gráfico 5.11: Comparações entre resultados da otimização do Rykel 48, para um enxame de 110 partículas, com e sem o operador Mudança de Configuração.

5.1.4 Tabelas e gráficos para o Rykel 48 utilizando enxames expurgados

Nesta série de experimentos foi utilizado o algoritmo da figura 4.7, para que o algoritmo ficasse com as características do PSO original.

5.1.4.1. Rykel 48 com enxame de 500 partículas.

Abaixo temos a tabela e o gráfico do enxame de melhor desempenho, obtido com as constantes w=0.7 c1=c2=1.8.

Iteração	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
Melhor Fitness	19555	17577	17223	16934	16663	16347	16347	16347	16343	16339
Fitness Média do Enxame	26831	25734	25828	27659	27757	27859	27336	27344	27143	27241

Tabela 5.23: Melhor resultado de um enxame de 500 partículas para o Rykel 48.

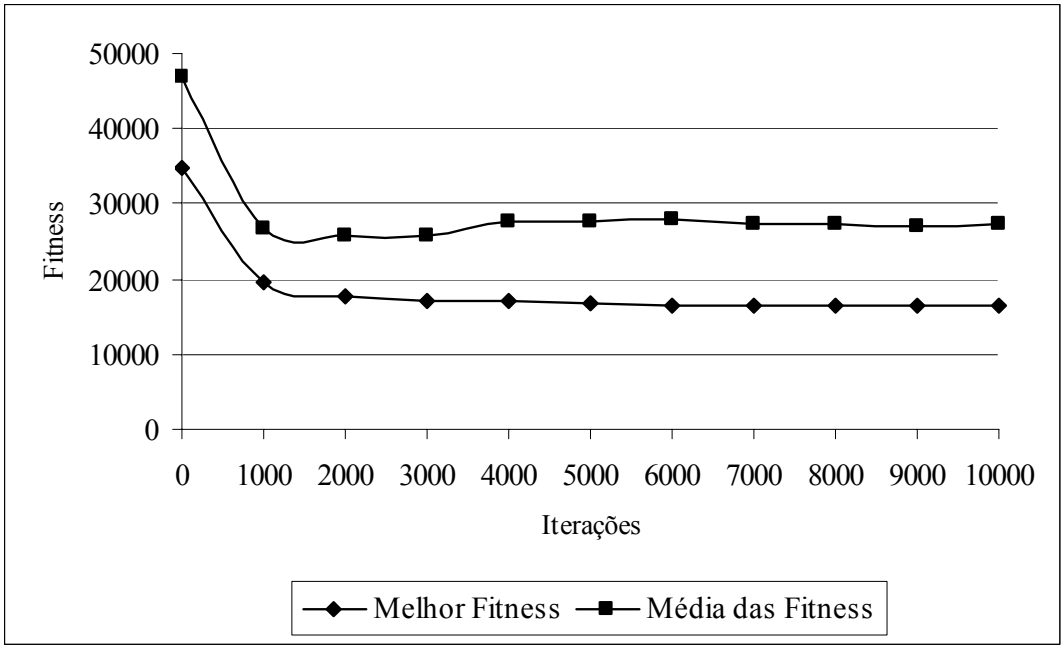


Gráfico 5.12: Melhor desempenho obtido na otimização do Rykel 48. Resultado para um enxame de 500 partículas, ao longo de 10.000 gerações.

Este experimento com enxames de 500 partículas para o Rykel 48 foi realizado com 10 sementes distintas utilizando-se as mesmas constantes, a fim de se verificar a robustez do método. Abaixo está a tabela com os valores médios de fitness a cada iteração indicada e a média das fitness médias de cada enxame.

Iteração	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
Melhor Fitness	20120	18665	18290	18151	18010	17854	17814	17675	17674	17639
Fitness Média do Enxame	29161	27536	27520	27606	27782	27825	27821	27845	27732	27749

Tabela 5.24: Resultados médios para 10 experimentos realizados, todos com as mesmas constantes, sementes distintas e enxames de 500 partículas para o Rykel 48.

O gráfico é mostrado abaixo:

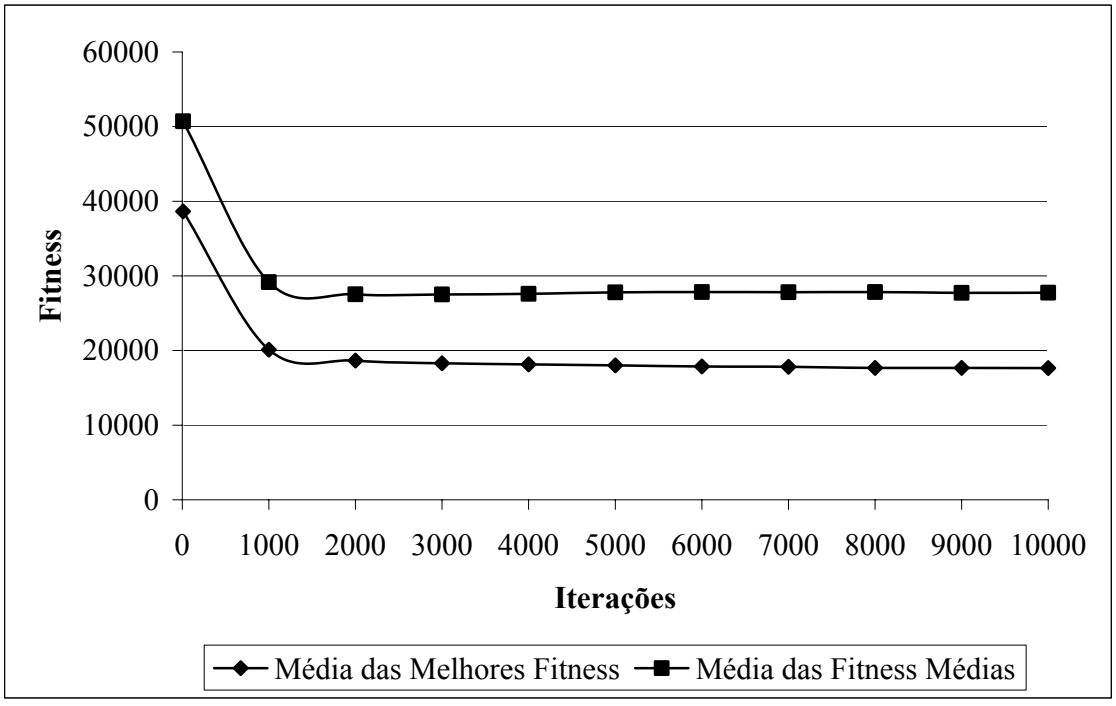


Gráfico 5.12: Resultados médios da otimização do Rykel 48, para 10 experimentos com enxames de 500 partículas.

5.1.4.2. Rykel 48 com enxame de 1000 partículas.

Abaixo temos a tabela e o gráfico do enxame de melhor desempenho, obtido com as constantes w=0.7 c1=c2=1.8.

Iteração	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
Melhor Fitness	16912	15672	15504	15504	15504	15504	15504	15504	15504	15504
Fitness Média do Enxame	23859	26035	25477	24792	24583	24347	24033	23886	23794	23497

Tabela 5.25: Melhor resultado de um enxame de 1000 partículas para o Rykel 48.

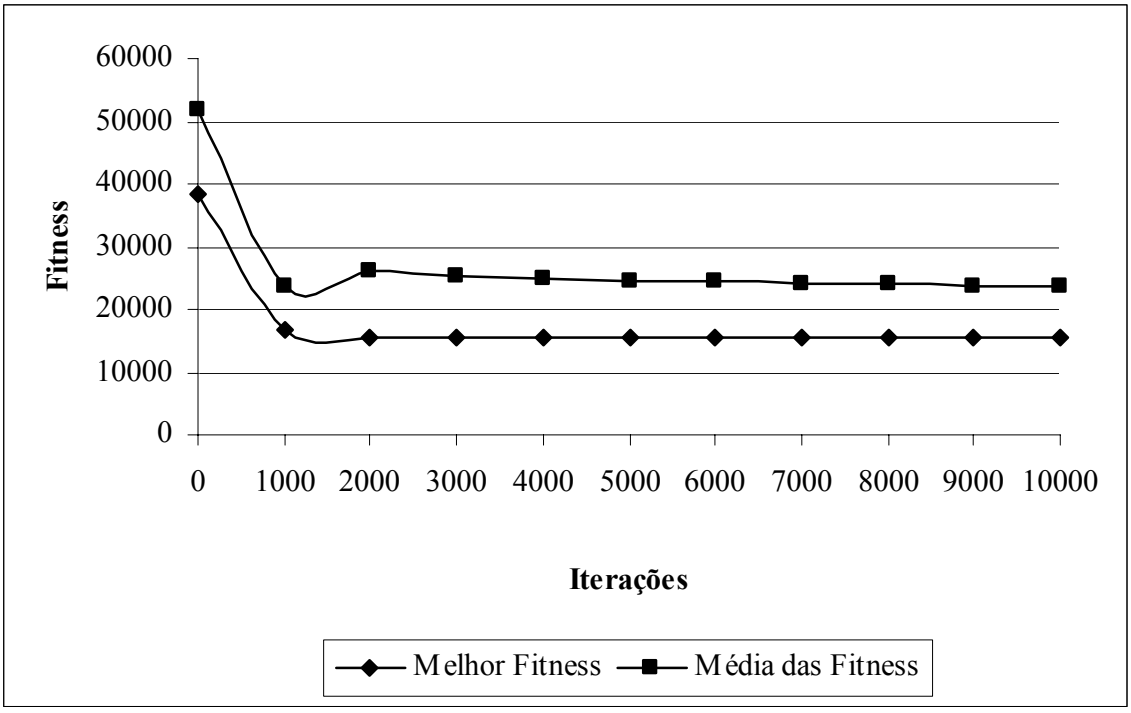


Gráfico 5.13: Melhor desempenho obtido na otimização do Rykel 48. Resultado para um enxame de 500 partículas, ao longo de 10.000 gerações.

Este experimento com enxames de 1000 partículas para o Rykel 48 foi realizado com 10 sementes distintas utilizando-se as mesmas constantes, a fim de se verificar a robustez do método. Abaixo está a tabela com os valores médios de fitness a cada iteração indicada e a média das fitness médias de cada enxame.

Iteração	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
Melhor Fitness	19242	17886	17483	17235	17033	16895	16874	16834	16812	16794
Fitness Média do Enxame	28529	27253	26630	27309	27277	27372	27230	27307	27214	27169

Tabela 5.26: Resultados médios para 10 experimentos realizados, todos com as mesmas constantes, sementes distintas e enxames de 1000 partículas para o Rykel 48.

O gráfico correspondente é:

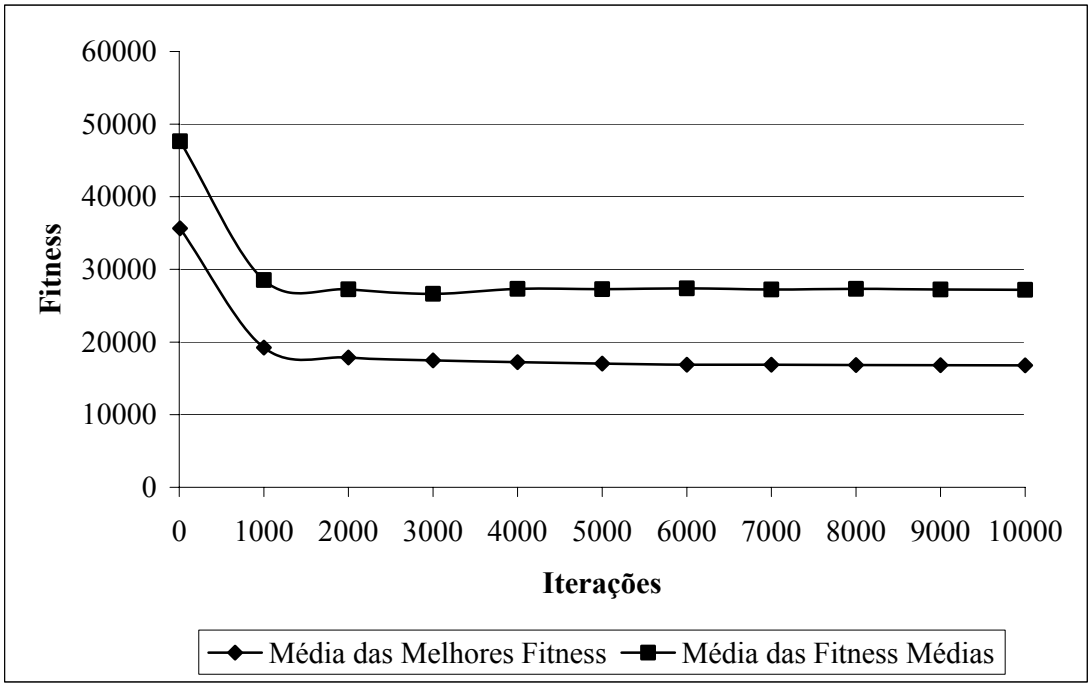


Gráfico 5.14: Resultados médios da otimização do Rykel 48, para 10 experimentos com enxames de 1000 partículas.

5.2. Comentários e Análise dos Resultados

Podemos observar algumas regularidades na obtenção da procura realizada pelos enxames. A partir dos resultados dos ensaios podem ser tecidos alguns comentários:

(i) Há homogeneidade entre os resultados.

Os conjuntos de dados são típicos e representativos, ou seja, outros experimentos nas mesmas condições e com sementes distintas conduzem a resultados similares. Os comportamentos dos enxames se repetem em um padrão.

(ii) Há uma relação entre a fitness média e melhor fitness.

Conforme a fitness da melhor partícula diminui, a fitness média diminui regularmente. Quando há convergência prematura, a fitness média fica estagnada ou até mesmo aumenta.

(iii) Influência do número de partículas no processo.

Os resultados das experiências mostram que, para o Oliver 30, o mínimo global é encontrado rápida e regularmente ou então são encontrados mínimos locais muito próximos dos globais. Existem casos em que os enxames não encontraram valores ótimos de maneira tão rápida como nos outros casos, entretanto não se descarta a possibilidade de serem encontrados bons resultados com um número maior de gerações. De um modo geral, nos testes realizados, para um enxame com um número de partículas menor do que 60 (no Oliver 30) ou 110 (no Rykel 48), há convergência muito demorada. Para enxames expandidos, tivemos um salto qualitativo na regularidade de obtenção de bons mínimos em

ambos os TSPs, com considerável redução de iterações, o que foi muito importante principalmente no Rykel 48.

(iv) Escolha de constantes.

A escolha das constantes foi feita com base no histórico de resultados obtidos em testes anteriores, onde diversas delas foram utilizadas. A seleção das constantes é um passo importante, pois é a partir das mesmas que a capacidade de busca do enxame será balanceada e cada partícula pode ter maior capacidade de exploração pela sua própria experiência ou pela experiência do enxame. Como será mencionado adiante, outros testes podem ser realizados no sentido de descobrir que constantes podem ser mais adequadas a cada problema e a cada tamanho de enxame. Entretanto é importante notar que, para o Rykel 48, as constantes aplicadas puderam ser levemente variadas (item 5.1.3.2) e mesmo assim o método não perdeu robustez, proporcionando resultados condizentes com os obtidos anteriormente.

(v) Operador de Mudança de Configuração

Este operador, utilizado no Rykel 48 para enxames reduzidos, seleciona as partículas que obtiveram os piores resultados em uma geração e muda drasticamente sua configuração, permitindo que elas simplesmente imitem a configuração da melhor partícula. Em princípio pode-se pensar que isto afetaria a diversidade do grupo. Como as mesmas não perdem sua experiência anterior, chegam a uma boa configuração sem, no entanto, saber como a melhor partícula a obteve. Daí ficam perdidas e procurando em diversas direções. Podemos inicialmente destacar que sua ausência piora significativamente os resultados. Já a sua utilização deve ser criteriosa com relação ao número de partículas envolvidas

no processo. Não é qualquer número de partículas invejosas que pode melhorar o resultado, havendo a possibilidade de que um número excessivo delas possa fazer o enxame ficar "perdido", comprometendo seu desempenho. A partir daí é necessário realizar estudos sobre a sua influência no método de busca.

(vi) Rapidez de convergência x número de gerações .

Apesar de não terem sido estipulados critérios de análises a partir do tempo de execução, notou-se que os resultados mostrados são obtidos em um tempo computacional extremamente rápido já que as operações feitas durante a execução são basicamente primitivas, como adição e multiplicação, não existindo procedimentos complexos ou demorados. Em contrapartida, para enxames reduzidos, o número de gerações é grande: no Oliver 30, chega-se a 200 gerações; no Rykel 48, chega-se a 50.000 gerações. O número de avaliações de fitness (*trials*) pode, então, para 80 partículas no Oliver 30, ser da ordem de 16.000. Já no Rykel 48, pode chegar a 6.000.000 para uma população de 120 partículas em 50.000 gerações como nos casos aqui apresentados, mas que não passam de 10 minutos de execução em um processador Intel Celeron 1,8GHz. Para enxames expandidos no Oliver 30, o melhor resultado obteve o mínimo global em uma média de 80 iterações, o que para 500 partículas resulta em 40.000 avaliações, enquanto, por exemplo, o Algoritmo Genético pode encontrar em 20.000 (MACHADO, 1999). Já para o Rykel 48, o melhor resultado mostra 1000 partículas levando 10000 iterações para chegar ao mínimo local de 15504, com 10^7 avaliações, mínimo comparável aos 15430 também obtido pelo Algoritmo Genético (AG) em 2.10^5 avaliações (MACHADO, 1999). O número de *trials*, como pode ser visto, ainda é grande, mas o método é recente e ainda existe muito

a ser explorado. Futuras adaptações ao problema de recarga devem levar este aspecto em consideração. Em contrapartida, outro fato importante que pode ser observado nos gráficos é que a otimização no Rykel é extremamente eficiente nas primeiras 2000 iterações, sendo que depois perde-se esta capacidade, sendo a convergência muito mais lenta. Entretanto, com 1000 partículas em 2000 iterações chega-se a 15672, também comparável ao AG, levando-se 2.10^6 iterações.

Com estes resultados, pudemos mostrar que a técnica PSORK apresenta resultados razoáveis para os problemas propostos. Este primeiro passo é fundamental para a otimização da recarga de combustível em um reator nuclear. Entretanto, para a efetiva migração da técnica para o problema da recarga, alguns fatores devem ser levados em consideração, como por exemplo, mais testes destinados a consolidar a técnica. Tais fatores e propostas de trabalhos futuros são mencionados no capítulo seguinte, que também apresenta as conclusões gerais desta dissertação.

Capítulo 6

Conclusões e Propostas de Trabalhos Futuros

O presente capítulo expõe as conclusões gerais desta dissertação e as propostas de trabalhos futuros com a técnica apresentada: o PSORK.

6.1. Comentários Gerais Sobre a Técnica de Otimização por Enxame de Partículas

A Otimização por Enxame de Partículas (PSO) é uma importante contribuição de KENNEDY e EBERHART (1995) que foi desenvolvida tendo como base os modelos de movimentos de cardumes de peixes e bandos de pássaros estudados nas áreas de Biologia e Ecologia. O movimento coletivo destes animais e a influência da individualidade no comportamento global abriram caminhos para esta técnica de otimização. O salto dado entre a simples representação do movimento coletivo e uma nova técnica de otimização de Inteligência Artificial foi comparar uma área de pouso com um ponto ótimo de uma função.

A partir de então, a técnica foi aplicada às mais variadas funções contínuas (PARSOPOULOS e VRAHATIS, 2002), mostrando-se uma importante ferramenta na otimização de problemas. O PSO é uma técnica evolutiva de IA que, enquanto metáfora, se situa entre os Algoritmos Genéticos e as Redes Neurais, já que processos evolutivos em populações levam milhares de anos, enquanto processos neurais podem transcorrer em milissegundos. Assim, a cognição social está entre estes dois pontos, interligando-

os, tanto em relação às escalas de tempo quanto ao efetivo desencadeamento dos processos. Por exemplo, uma mutação genética pode modificar o comportamento de um indivíduo dentro de um grupo. Em seguida, se isto trouxer alguma vantagem adaptativa, seus descendentes apresentarão possibilidade de herdá-la e mesmo outros indivíduos da mesma geração poderão imitar ou aprender tal comportamento, aumentando a disseminação desta característica e conseqüentemente as probabilidades de sobrevivência do grupo. Assim haverá mudança em aspectos sociais do grupo. Em todo este processo, informações serão processadas neurologicamente nos componentes do grupo, liberando substâncias cerebrais em maiores ou menores proporções, trazendo mudanças em suas quantidades. Logo, alterações genéticas combinadas com o fator social podem levar a evoluções neurais nas espécies. E o contrário também pode ocorrer. Como exemplo disto, temos as recompensas e punições ao longo da vivência de cada indivíduo, com reações na química cerebral gerando alterações nos comportamentos individuais, que poderão ser imitados ou aperfeiçoados por outros integrantes do grupo. Em longo prazo, a repetição de atividades que propiciem a sobrevivência global pode levar a heranças de características pelas gerações seguintes bem como a mutações. Assim, podemos dizer que os efeitos dos processos genéticos e neurais das espécies apresentam relações com os processos sociais.

Levando em conta aspectos teóricos e o estado da arte da metáfora do aprendizado social e sua modelagem como técnica de IA, procuramos neste trabalho, estudar a técnica de PSO propondo sua aplicação a problemas combinatórios com vistas à otimização da recarga de combustível do núcleo de um reator nuclear. O problema de encontrar o melhor posicionamento de elementos combustíveis do núcleo de um reator tem alto grau de complexidade: é multimodal, não-linear e as avaliações de sua função

objetivo demandam considerável tempo computacional. Além disso, trata-se de um problema combinatório NP-difícil, de acordo com a teoria de complexidade computacional. Segundo KENNEDY e EBERHART (1995), o PSO foi introduzido como "um método para otimização de funções contínuas e não-lineares". Entretanto, é pouco usual encontrar modelagens para funções descontínuas em problemas combinatórios. Existem, porém, algumas delas, mas seus resultados não se mostram satisfatórios para problemas com a magnitude da recarga de combustível nuclear. Esta foi uma das motivações para a realização deste trabalho, do qual apresentamos as conclusões gerais na seção seguinte.

6.2. Conclusões Gerais

Para estudarmos o PSO na busca da solução do problema de recarga tivemos que adaptar a técnica. Com base na aplicação da técnica feita por SALMAN et al. (2002) ao problema TAP, pudemos acrescentar a utilização do modelo de codificação Random Keys, dando origem ao PSORK, permitindo a resolução de outro problema combinatório: o TSP. Os TSPs Oliver 30 e Rykel 48 aqui utilizados e em cujas soluções o PSORK foi aplicado, são importantes para estudos de técnicas de solução do problema de recarga pois a ordem de grandeza do número de soluções é compatível com o problema da recarga, segundo as modelagens existentes para os cálculos de Física de Reatores. Além disto, existe a vantagem de o TSP possuir uma função objetivo extremamente simples e cujo tempo de avaliação é rápido, contrastando com a avaliação do problema de recarga. Já que a versão de otimização do TSP também é NP-difícil (PAPADIMITRIOU e STEIGLITZ, 1982, p. 398), então a resolução do TSP pode servir

como referência para a aplicação de técnicas ao problema da recarga. Além disto, vários trabalhos recentes mostram aplicações de técnicas de otimização de IA bem-sucedidas na aplicação ao TSP que na recarga também forneceram bons resultados, sendo um indício de que uma dada técnica, ao resolver um TSP com um número de combinações similar, pode potencialmente fornecer bons resultados na otimização da recarga do combustível nuclear (CHAPOT, 2000; LIMA, 2005; MACHADO, 2005).

Neste trabalho apresentamos duas versões do PSORK: uma delas aplicada ao TSP simétrico Oliver 30 e outra ao TSP assimétrico Rykel 48, obtidas depois de muitos testes realizados. A primeira faz com que as partículas se movam em um espaço de busca de vetores de números reais, mas a partir da comparação de vetores de números inteiros. Neste caso, as partículas usam a equação (19) para comparar ordens de visitação de cidades diretamente. Na segunda, as posições das partículas são atualizadas a partir da comparação das próprias chaves randômicas. Neste outro caso, utilizam-se as equações (21) e (23), respectivamente para enxames reduzidos e enxames expandidos. Inicialmente, para ambas as versões foram desenvolvidas sub-rotinas que ajudassem a manter a diversidade dos enxames. E para a solução do Rykel 48 ainda foram estabelecidos: (a) um critério diferenciado para a procura feita pelas melhores partículas, no caso, um pouco mais refinada do que para o restante do enxame e (b) mudança de configuração das piores partículas, para busca de soluções com configurações parecidas com as melhores partículas. Posteriormente, no caso do TSP Rykel 48, ao usarmos enxames com 500 e 1000 partículas conforme realizado por CANEDO (2005), alcançamos os resultados já mostrados no capítulo 5, e o mais importante, sem o auxílio das sub-rotinas destinadas a manter a diversidade e sem os procedimentos (a) e (b) citados acima, aproximando o modelo proposto ao PSO

original, havendo maior consistência, regularidade e possibilidade de comparação com outros métodos de otimização como PBIL e GA (MACHADO, 1999).

Sendo assim, seguem as conclusões gerais deste trabalho, com base nos resultados obtidos no capítulo 5:

(i) há possibilidade de utilização da técnica de PSO na otimização de problemas combinatórios utilizando o Random Keys para codificação/decodificação na busca.

(ii) o PSORK mostrou-se satisfatório principalmente se for levado em conta que a otimização do TSP com o PSO foi apenas conseguida para problemas simétricos de 14 cidades, com aproximadamente $3{,}1.10^{9}$ soluções (WANG et al., 2003) e 17 cidades, com aproximadamente $1{,}0.10^{13}$ soluções (CLERC, 2004). Neste trabalho, mostramos a otimização do TSP simétrico Oliver 30, com aproximadamente $4{,}4.10^{30}$ assimétrico de 48 cidades e do TSP assimétrico Rykel 48, com aproximadamente $2{,}5.10^{59}$ soluções possíveis.

(iii) o PSORK forneceu resultados homogêneos e comparáveis a outros métodos de otimização já consolidados (para o Oliver 30 o PSO obtém o mínimo como no GA; para o Rykel chega-se a 15504, comparáveis aos 15430 obtidos pelo GA), podendo inclusive haver maiores estudos no sentido de refinamento da técnica para que o número total de avaliações seja menor, reduzindo ainda mais o tempo computacional. Já que cada avaliação de função objetivo da recarga necessita de considerável tempo computacional, é desejável que a otimização proceda de modo a utilizar o menor número global possível de avaliações.

(iv) levando em conta o exposto nos itens (i), (ii), (iii) acima, podemos finalmente afirmar que há possibilidade de utilização do PSORK no problema da recarga de combustível de reatores nucleares.

6.3. Propostas de Trabalhos Futuros

Inicialmente poderá ser realizado um estudo pormenorizado da sensibilidade do sistema em relação às constantes utilizadas nas fórmulas. Há possibilidade de estudar também o efeito da modificação das constantes com relação a um maior ou menor número de partículas no enxame.

Em virtude de terem sido apresentadas duas versões da técnica, uma para o TSP simétrico Oliver 30 e outra para o assimétrico Rykel 48, também poderão ser estudadas adaptações da técnica e comparações de eficiência entre as duas modalidades, dependendo do número de cidades e simetria/assimetria dos problemas.

Pode-se futuramente testar variações com auxílio das técnicas de vizinhanças, nichos e parametrização, bem como estudos de implementação do algoritmo em processamento distribuído.

É essencial também, depois de um refinamento da técnica e aumento de sua eficiência, aplicá-la diretamente ao problema da otimização da recarga de combustível

em reatores nucleares, para comparação com resultados obtidos por outros métodos de otimização.

Finalmente, assim como há possibilidade de usar redes neurais para desenvolvimento de sistemas especialistas conexionistas, poderá ser viabilizado um estudo de como adaptar a metáfora do aprendizado social a um sistema especialista.

6.4. Considerações Finais

Com o trabalho desenvolvido, pudemos mostrar que a relativamente recente técnica de PSO pode alcançar bons resultados, comparáveis aos obtidos por outros métodos de otimização em IA, com o auxílio do Random Keys.

Desta forma, temos indícios de que a metáfora do aprendizado social, implementada através do PSO, pode começar a ser aplicada de maneira eficiente em um novo e amplo campo de pesquisas à sua frente, que é a otimização de problemas combinatórios. E isto contempla também os problemas combinatórios de interesse na Engenharia Nuclear, em particular, o da recarga de combustível em reatores nucleares.

Apêndice A

Esquema do Núcleo de um Reator PWR

A figura abaixo representa um núcleo de Reator a Água Pressurizada (PWR, Pressurized Water Reactor), contendo 121 elementos combustíveis, como o da Usina Nuclear de Angra I.

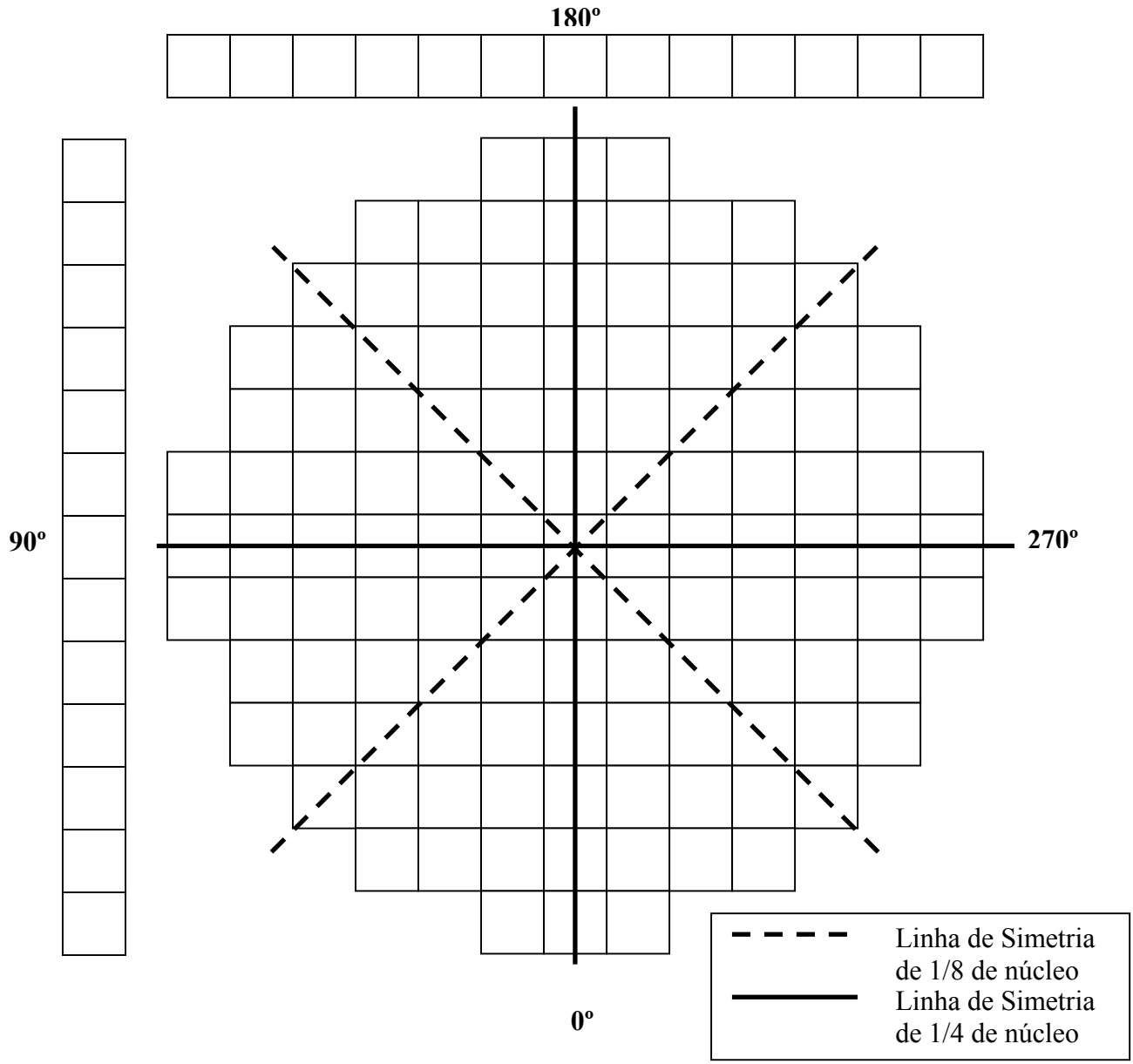


Fig. A.1: Representação de um núcleo de reator tipo PWR e linhas de simetria.

REFERÊNCIAS BIBLIOGRÁFICAS

BEAN, James C. Genetics *Algorithms and Random Keys for Sequencing and Optimization*. ORSA Journal of Computing, Vol. 6, No. 2, 1994.

BURIOL, L. S. *Algoritmo Memético para o Problema do Caixeiro Viajante Assimétrico como Parte de Framework para Algoritmos Evolutivos.* Tese de Mestrado. Campinas: DENSIS/FEE/UNICAMP, 2000.

CHAPOT, J. L. C. *Otimização Automática de Recargas de Reatores a Água Pressurizada Utilizando Algoritmos Genéticos.* Tese de Doutorado. Rio de Janeiro: COPPE/UFRJ, 2000.

CLERC, Maurice. *Discrete Particle Swarm Optimization, Illustrated by the Traveling Salesman Problem.* France Télécom Recherche & Développement, 2004.

COOK, S. A. *The Complexity of Theorem Proving Procedures*. Proc. 3rd ACM Symp. On the Theory of Computing, ACM (1971), pp. 151-158.

DANTZIG, G. B. *On the significance of solving linear programming problems with com integer variables*. Econometrica 28, (pp. 30-44), 1960.

DORIGO, Marco, MANIEZZO, Vittorio e COLORNI, Alberto. *The Ant System: Optimization by a colony of cooperation agents*. IEEE Transactions on systems, Man and Cybernetics – Part B, Vol. 26, No. 1, pp.1-13. 1996.

DORIGO, Marco e GAMBARDELLA, Luca Maria. *A Study of Some Properties of Ant-Q*. Proceedings of PPSN IV – Fourth International Conference on Parallel Problem Solving From Nature, H.–M. Voigt, W. Ebeling, I. Rechenberg and H.–S. Schwefel (Eds.), Springer-Verlag, Berlin, pp. 656–665, 1996.

DUDERSDADT, James J. e HAMILTON, Louis J. *Nuclear Reactor Analysis.* p. 566-580. EUA: John Wiley and Sons, Inc., 1976.

EBERHART, R. C. e KENNEDY, J. *A New Optimizer Using Particles Swarm Theory*. Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan) IEEE Service Center, Piscataway, NJ: 39-43, 1995.

EBERHART, R. C. e KENNEDY, J. *A discrete binary version of the particle swarm algorithm*. Conference on Systems, Man and Cybernetics, pp. 4104-4109, 1997.

GAREY, M. R. e JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-completeness.* São Francisco: W. H. Freeman & Company, Publishers, 1979.

GOLDBERG, David E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts, EUA: Addison-Wesley Publishing Company, Inc., 1989.

HEPPNER, F. e GRENANDER, U. *A Stochastic Nonlinear Model for Coordinated Bird Flocks*. In Krasner, S., Ed., The Ubiquity of Chaos. Washington, EUA: AAAS Publications, 1990.

HOFFMAN, A. J. e WOLFE, P. *History. In:* LAWLER, E. L., LENSTRA, J. K., KAN, A. H. G. R., SHMOYS, D. B. (Org.). *The Traveling Salesman Problem:* a guided tour of combinatorial optimization. 4ª. Ed. Wiltshire, Grã-Bretanha: John Wiley & Sons, 1985. (pp. 1-15)

KENNEDY, J. e EBERHART, R. C. *Swarm intelligence*. San Diego, EUA: Academic Press, 2001.

LIMA, A. M. M. de. *Modelo de Ilhas para Implementação do Algoritmo Evolucionário de Otimização PBIL*. Tese de Mestrado. Rio de Janeiro: COPPE/UFRJ, 2000.

LIMA, A. M. M. de. *Recarga de Reatores Nucleares Utilizando Redes Conectivas de Colônias Artificiais.* Tese de Doutorado. Rio de Janeiro: COPPE/UFRJ, 2005.

MACHADO, M. D. *Um novo algoritmo evolucionário com aprendizado LVQ para a otimização de problemas combinatórios como a recarga de reatores nucleares*. Tese de Mestrado. Rio de Janeiro: COPPE/UFRJ, 1999.

MACHADO, M. D. *Algoritmo Evolucionário PBIL Multiobjetivo Aplicado ao Problema de Recarga de Reatores Nucleares*. Tese de Doutorado. Rio de Janeiro: COPPE/UFRJ, 2005.

McFARLANE, A. F. *Topical Report Power Peaking Factors,* WCAP – 7912 – L, Westinghouse Electric Corporation, 1972.

MEDEIROS, J. A. C. C. *Enxame de Partículas como Ferramenta de Otimização em Problemas Complexos da Engenharia Nuclear*. Tese de Doutorado. Rio de Janeiro: COPPE/UFRJ, 2005.

MENESES, A. A. de M. e SCHIRRU, R. *Particle Swarm Optimization Aplicado ao Problema Combinatório com Vistas à Solução do Problema de Recarga em um Reator Nuclear*. International Nuclear Atlantic Conference (INAC) 2005. XIV Encontro Nacional de Física de Reatores (ENFIR). Agosto de 2005.

PAPADIMITRIOU, Christos H. e JOHNSON, D. S. *Computational Complexity. In:* LAWLER, E. L., LENSTRA, J. K., KAN, A. H. G. R., SHMOYS, D. B. (Org.). *The Traveling Salesman Problem:* a guided tour of combinatorial optimization. 4ª. Ed. Wiltshire, Grã-Bretanha: John Wiley & Sons, 1985. (pp. 37 a 85)

PAPADIMITRIOU, Christos H. e STEIGLITZ, Kenneth. *Combinatorial Optimization*. New Jersey: Prentice-Hall, Inc. 1982.

PARSOPOULOS, K. E. e VRAHATIS, M. N. *Recent aproaches to global optimization problems through Particle Swarm Optimization.* Natural Computing 1 (pp. 235-306). Países Baixos: Kluwer Academic Publishers, 2002.

POMEROY, PAUL. *An Introduction to Particle Swarm Optimization.* http:// www.adaptiveview.com/articles/ipsoprnt.htm. Acessado em 20Fev2004.

SALMAN, Ayed, AHMAD, Imtiaz e AL-MADANI, Sabah. *Particle Swarm Optimization for Task Assignment Problem*. Microprocessors and Microsystems, Volume 26, Issue 8, 10 November 2002, pages 363-371.

SANTOS, D. R. G. *Algoritmos Genéticos Paralelos como Ferramenta para Recarga de Reatores Nucleares*. Tese de Mestrado. Rio de Janeiro: COPPE/UFRJ, 1998.

SECREST, B. R. e LAMONT, G. B. *Visualizing Particle Swarm Optimization – Gaussian Particle Swarm Optimization.* Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003), Indianapolis, Indiana, EUA, 2003 (pp. 198-204).

TODREAS, Neil E. e KAZIMI, Mujid S. *Nuclear Systems I:* Thermal Hydraulic Fundamentals. Estados Unidos da América: Hemisphere Publishing Corporation, 1990.

TRELEA, I. C. *The particle swarm optimization algorithm: convergence analysis and parameter selection.* Information Processing Letters 85, 2003 (pp. 317-325).

TSPLIB. Disponível em <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>. Último acesso em 22 Set 2005.

WANG, Kang-Ping, HUANG, Lan, ZHOU, Chun-Guang, PANG, Wei. *Particle Swarm Optimization for Traveling Salesman Problem*. Proc. of the Second International

Conference on Machine Learning and Cybernetics, Xi'an, 2-5 November 2003. Pages 1583-85. 2003.

WILSON, E. O. *Sociobiology: The New Synthesis.* Cambridge, MA: Belknap Press, 1975.

YIN, Peng-Yeng. *A discrete particle swarm algorithm for optimal polygonal approximation of digital curves.* Journal of Visual Communication and Image Representation, 2004, p. 241-260.