Viviane Müller

OTIMIZAÇÃO DE *LAYOUTS* INDUSTRIAIS ATRAVÉS DO MÉTODO ENXAME DE PARTÍCULAS

Dissertação apresentada ao Programa de Pós-Graduação Sistemas e Processos Industriais – Mestrado, Área de Concentração em Controle e Otimização de Processos Industriais, Universidade de Santa Cruz do Sul, como requisito parcial para obtenção do título de Mestre em Sistemas e Processos Industriais.

Orientador: Prof. Dr. João Carlos Furtado

Co-orientador: Prof. Dr. Rafael Ramos dos Santos

Santa Cruz do Sul, janeiro de 2007

COMISSÃO EXAMINADORA

Titulares

Prof. Dr. João Carlos Furtado – Orientador

Prof. Dr. Rafael Ramos dos Santos – Co-orientador

Prof. Dr. Geraldo Lopes Crossetti – UNISC

Prof. Dr. Denis Borenstein – UFRGS

AGRADECIMENTOS

Agradeço a Deus pela vida e pelas bênçãos recebidas.

Agradeço aos melhores pais do mundo, Arno e Rosvita, por tudo que fizeram por mim.

Agradeço aos meus irmãos, Vanessa e João Henrique, pela amizade sincera e eterna.

Agradeço ao meu “noivorido”, Rodrigo, pelo seu amor.

Agradeço ao meu orientador, prof. Dr. João Carlos Furtado, pela ajuda e disponibilidade.

Agradeço ao meu co-orientador, prof. Dr. Rafael Ramos dos Santos, pelo reconhecimento.

Agradeço ao Coordenador do Programa de Pós-Graduação em Sistemas e Processos Industriais, prof. PhD. Marco Flôres Ferrão, pelas oportunidades.

Agradeço aos demais professores doutores do Mestrado em Sistemas e Processos Industriais: Geraldo Lopes Crossetti, Rejane Frozza, Jacques Nelson Corleta Schreiber, Rolf Fredi Molz e Ruben Edgardo Panta Pazos. Muito obrigada pelo bom humor, pelos ensinamentos e bons exemplos.

Agradeço ao *expert* Jaime Neis, pela ajuda na programação.

Agradeço às minhas amigas e amigos, em especial à Andrea Konzen pelo apoio e à Karen Santorum pelas traduções realizadas e pela prontidão em me atender. Obrigada pelo carinho!

Agradeço às secretárias, Cláudia e Janaina, pela presteza e atenção.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Capes pelo auxílio financeiro.

Agradeço a todos que torceram para que eu me tornasse **Mestre!**

Salmo 90

1 Senhor, tu tens sido o nosso refúgio de geração em geração.

2 Antes que nascessem os montes, ou que tivesses formado a terra e o mundo, sim, de eternidade a eternidade tu és Deus.

3 Tu reduzes o homem ao pó, e dizes: Voltai, filhos dos homens!

4 Porque mil anos aos teus olhos são como o dia de ontem que passou, e como uma vigília da noite.

5 Tu os levas como por uma torrente; são como um sono; de manhã são como a erva que cresce;

6 de manhã cresce e floresce; à tarde corta-se e seca.

7 Pois somos consumidos pela tua ira, e pelo teu furor somos conturbados.

8 Diante de ti puseste as nossas iniqüidades, à luz do teu rosto os nossos pecados ocultos.

9 Pois todos os nossos dias vão passando na tua indignação; acabam-se os nossos anos como um suspiro.

10 A duração da nossa vida é de setenta anos; e se alguns, pela sua robustez, chegam a oitenta anos, a medida deles é canseira e enfado; pois passa rapidamente, e nós voamos.

11 Quem conhece o poder da tua ira? e a tua cólera, segundo o temor que te é devido?

12 Ensina-nos a contar os nossos dias de tal maneira que alcancemos corações sábios.

13 Volta-te para nós, Senhor! Até quando? Tem compaixão dos teus servos.

14 Sacia-nos de manhã com a tua benignidade, para que nos regozijemos e nos alegremos todos os nossos dias.

15 Alegra-nos pelos dias em que nos afligiste, e pelos anos em que vimos o mal.

16 Apareça a tua obra aos teus servos, e a tua glória sobre seus filhos.

17 Seja sobre nós a graça do Senhor, nosso Deus; e confirma sobre nós a obra das nossas mãos; sim, confirma a obra das nossas mãos.

Viver é muito mais do que fazemos habitualmente.

RESUMO

O problema de *layout* de facilidades é antigo e consiste, basicamente, em encontrar a melhor forma de dispor as facilidades (departamentos, máquinas, etc.) dentro de um campo de aplicação (indústria, hospital, banco, etc.). Embora o problema seja fácil de entender, a sua resolução é difícil devido à complexidade computacional. Diferentes métodos foram empregados na sua resolução e continua-se a busca por métodos mais eficientes devido a importância científica e econômica. Este trabalho propõe o uso do método Enxame de Partículas (*Particle Swarm Optimization* - PSO) na otimização do problema de *layout*. Enxame de Partículas é uma heurística evolutiva baseada em população (enxame), formada por indivíduos (partículas), cuja evolução se dá por meio da velocidade. É um algoritmo de conceito simples, fácil implementação, robustez para controlar parâmetros e eficiência computacional durante o processo de otimização. Cada partícula possui uma velocidade que permite a ela percorrer o espaço de soluções e uma posição nesse espaço. A cada iteração, e para cada partícula, a velocidade é atualizada de acordo com a velocidade anterior, somada à parte cognitiva da fórmula, que representa o conhecimento, e à parte social, que representa a colaboração entre as partículas. A nova posição da partícula é determinada pela soma da sua posição atual e a nova velocidade. Este trabalho apresenta duas abordagens, desenvolvidas a partir do algoritmo enxame de partículas original, para resolver o problema de *layout*. A segunda abordagem apresentou melhores resultados e, comparando-se com outros resultados encontrados na literatura, pode-se comprovar a eficiência do método Enxame de Partículas para o problema de *layout*, para todas as instâncias testadas.

Palavras-chave: Problema de *layout* de facilidades, Enxame de Partículas, modelo Attractor-Repeller, otimização.

ABSTRACT

The facility layout problem is old and it consists, basically, in finding the best way of arrangement of the facilities (machines, departments, etc.) inside of an application field (industry, hospital, banc, etc.). Although the problem is easy to understand, its solution is hard due to the computational complexity. Different methods were used in its resolution and the search for more efficient methods due to the scientific and economic importance still takes place. This work proposes the use of the method of Particle Swarm Optimization - PSO to optimize the layout problem. Particle Swarm is an evolutionary metaheuristic based in population (swarm), formed by individuals (particles), whose evolution happens through the speed. It is an algorithm of simple concept, easy implementation, robustness to control parameters and computational efficiency during the optimization process. Each particle possesses a speed that allows it to travel the space of solutions and a position in that space. To each iteration, and for each particle, the speed is updated in agreement with the previous speed added to the cognitive part of the formula, that represents the knowledge and to the social part that represents the cooperation among the particles. The new position of the particle is determined by the sum of its current position and the new speed. The present work presents two approaches, developed from the algorithm particles swarm to solve the facility layout problem. The second approach presented better results and compared to other results found in the literature it is possible to prove the efficiency of the method Particle Swarm for the facility layout problem for all the tested instances.

Keywords: Facility layout problem, Particle swarm, Attractor-Repeller model, optimization.

LISTA DE SÍMBOLOS E SIGLAS

i, j – facilidades/partículas

d_{ij} – distância entre as facilidades i e j

x_i – posição da partícula i em relação ao eixo x

y_i – posição da partícula i em relação ao eixo y

H_{ij} – razão de aproximação entre as facilidades i e j

a_{ij} – adjacência entre as facilidades i e j

t_{ij} – distância-alvo entre as facilidades i e j

α – parâmetro que controla a sobreposição das facilidades

nl – coeficiente de não linearidade

G – conjunto de seqüência de operadores gerados da troca de um operador

r_i – raio da facilidade i

N – número de facilidades

n – número de partículas

c_{ij} – custo entre as facilidades i e j

P – constante de penalidade

wi – largura da facilidade

he – altura da facilidade

q_i – centro da facilidade i

v_i – velocidade atual da partícula i

c_1, c_2 – parâmetros de confiança

rand() – função aleatória

$pbest_i$ – melhor posição que a partícula i já obteve durante a busca

$gbest$ – melhor posição encontrada pelas partículas no enxame

c_{1ini} – valor inicial para o parâmetro de confiança cognitivo

c_{1fin} – valor final para o parâmetro de confiança cognitivo

c_{2ini} – valor inicial para o parâmetro de confiança social

c_{2fin} – valor final para o parâmetro de confiança social

R – número de iterações

it – iteração atual

s – número de *layouts*

V_{gbest} – velocidade próxima à melhor posição do enxame

V_{pbesti} – velocidade próxima à melhor posição da partícula

v_{max} – limitador de velocidade

w – componente inercial

w_{ini} = valor inicial para o coeficiente de inércia

w_{fin} = valor final para o coeficiente de inércia

k – fator de constrição

c_{3r} – parâmetros de confiança na iteração it

$gbest_i$ – posição da partícula i, quando obtida a melhor avaliação global do enxame

C – centróide do enxame

c_{3ini} = valor inicial para o parâmetro de atração da partícula pelo grupo

c_{3fin} = valor final para o parâmetro de atração da partícula pelo grupo

($pbest_i - x_i$) – distância entre a melhor posição já encontrada pela partícula i e a posição atual dessa mesma partícula

($gbest_i - x_i$) – distância entre a posição da partícula i quando o enxame atingiu o ótimo global e a posição atual dessa partícula.

AVOLI – ferramenta computacional chamada “Ambiente Visual para Otimização de *Layout* Industrial”

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

Desde a década de 60 vários estudos vem sendo realizados para encontrar a melhor solução para o problema de *layout*, que consiste, basicamente, em encontrar a melhor disposição de departamentos ou máquinas (facilidades) numa indústria, no caso de *layout* de facilidades ou a melhor disposição dos componentes e das interconexões entre eles, no caso de *layout* de circuitos eletrônicos, por exemplo.

O problema é visto como um problema de otimização combinatória, pertencente à classe de problemas não-determinísticos polinomiais completos, ou seja, não existe um algoritmo polinomial conhecido que encontre uma solução ótima em tempo computacional aceitável (SAHNI e GONZALEZ, 1976). Neste caso, pode-se usar heurísticas que são direcionadas à otimização global de um problema, capazes de obter soluções aproximadas.

Nas últimas décadas, surgiram vários procedimentos enquadrados como heurísticas na solução de diversos problemas altamente combinatórios, tais como: *Simulated Annealing* (KIRKPATRICK *et al*, 1983), Busca Tabu (GLOVER, 1989a; 1989b), *Greedy Randomized Search Procedure* – GRASP (FEO e RESENDE, 1995), *Variable Neighbourhood Search* – VNS (MLADENOVIC, 1995; MLADENOVIC e HANSEN, 1997), Algoritmos Genéticos (HOLLAND, 1975), Colônia de Formigas (DORIGO, 1992) e Enxame de Partículas (KENNEDY e EBERHART, 1995).

O método de otimização por Enxame de Partículas (*Particle Swarm Optimization* – PSO) foi inicialmente proposto por Kennedy e Eberhart (1995) e inspirado no comportamento de agentes sociais. Na natureza, esse comportamento pode ser observado em bandos de

pássaros, enxames de abelhas e cardumes de peixes, por exemplo. O modelo computacional é baseado na população, onde os agentes, chamados de partículas, mudam sua posição (estado) no espaço de busca, de acordo com a própria experiência e a experiência das partículas vizinhas que constituem o enxame.

Apesar de relativamente recente, o PSO tem recebido muita atenção pela sua simplicidade, robustez e eficiência e será aplicado neste trabalho para procurar obter melhores resultados na resolução do problema de *layout*.

1.1 Motivação

Poucas heurísticas, já aplicadas ao problema de *layout* na tentativa de encontrar uma solução ótima ou quase-ótima, podem ser consideradas eficientes, pois não cumprem o seu propósito em tempo de processamento e/ou memória aceitável. Por esta razão e, principalmente, pela importância da aplicabilidade de *layout* de qualidade na indústria, este trabalho verifica a eficiência do método enxame de partículas, ainda pouco explorado para este problema.

Para manterem-se competitivas e eficientes na sua área de atuação, as indústrias necessitam que a organização e posterior reorganização do *layout* sejam atividades constantes, devido, principalmente, à evolução tecnológica que produz novas máquinas e equipamentos, tornando modelos e métodos obsoletos. Sob o aspecto econômico há uma considerável redução de custos na minimização das interconexões dos departamentos para a indústria que adota um *layout* planejado e coerente (MOURA *et al*, 2002).

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é verificar a eficiência do método enxame de partículas na otimização do problema de *layout* de facilidades.

1.2.2 Objetivos específicos

Como objetivos específicos cita-se:

- Estudar o problema de *layout*;
- Identificar as características do método enxame de partículas;
- Avaliar os resultados obtidos no que diz respeito à aplicabilidade do método ao problema;
- Comparar os resultados da aplicação do método enxame de partículas a outros resultados obtidos para esse problema.

1.3 Metodologia

Este trabalho se origina de uma pesquisa exploratória, de uma pesquisa descritiva e de uma pesquisa experimental (SANTOS, 2000). Inicialmente, foi realizada uma busca exaustiva por bibliografia, nacional e, principalmente, internacional, já que as pesquisas sobre a técnica de otimização por enxame de partículas, o problema de *layout* e as abordagens para tentar resolvê-lo, continuam recentes no Brasil.

Ainda na fase exploratória, foi feito um levantamento bibliográfico de pesquisas já realizadas. Este material coletado foi utilizado na fase experimental do processo quando, a partir dos resultados obtidos com este trabalho, foi possível a comparação a fim de verificar a eficiência da técnica aplicada a este problema.

Após, com base nas leituras feitas a partir do material proveniente da pesquisa exploratória, foram relatadas as características do problema de *layout* e da técnica de otimização por enxame de partículas, a ser utilizada neste trabalho, o que constitui a pesquisa descritiva. Em seguida foi definida a modelagem do problema.

O protótipo foi desenvolvido utilizando a Linguagem de Programação Orientada a Objetos, Java, que é uma linguagem concisa, portável, independente de plataforma e com vários recursos multimídia e componentes de interface gráfica. O computador utilizado para o desenvolvimento do protótipo e os inúmeros testes realizados foi um Pentium IV, de 2.8 GHz e 512 MB de memória RAM.

Os resultados obtidos com a aplicação do método enxame de partículas ao problema de *layout* foram comparados com os resultados obtidos por Nugent (1968), autor das instâncias utilizadas e testadas neste trabalho e Anjos e Vannelli (2002), que desenvolveram o modelo *Attractor-Repeller*.

1.4 Estrutura da Dissertação

No próximo capítulo será descrito o problema de *layout*, o que é necessário para a contextualização e compreensão deste trabalho e serão apresentados os principais métodos utilizados para modelar este problema.

O capítulo 3 apresenta algumas das heurísticas já utilizadas na tentativa de resolução do problema de *layout*, suas principais características e resultados obtidos, o que é essencial para posterior comparação com os resultados obtidos neste trabalho. O capítulo 4 descreve o método enxame de partículas.

Nos capítulos 5 e 6 são apresentadas duas abordagens desenvolvidas para resolver o problema de *layout* utilizando o método enxame de partículas. Na primeira abordagem, cada facilidade é representada por uma partícula e o *layout* é o enxame e na segunda, vários *layouts* são utilizados para se obter um *layout* de qualidade. Os resultados obtidos são mostrados no final de cada capítulo. Finalmente, no último capítulo são apresentadas conclusões do trabalho desenvolvido.

2 O PROBLEMA DE *LAYOUT*

2.1 Um problema combinatorial

Durante a Segunda Guerra Mundial, como resultado de estudos realizados por equipes interdisciplinares de cientistas contratados para resolver problemas militares de ordem estratégica e tática (ANDRADE, 1998), surgiu a Pesquisa Operacional (PO).

PO é uma área multidisciplinar e caracterizada pela análise de decisões e pelo uso de técnicas e métodos científicos com o objetivo de determinar a melhor utilização dos recursos limitados e otimizar operações. Uma das subáreas da pesquisa operacional é a otimização combinatória.

A otimização combinatória está relacionada à ciência da computação – uma vez que, para obter soluções de qualidade, torna-se necessária a implementação em computador – e objetiva encontrar, através da maximização ou minimização de uma função, uma melhor solução dentro de um número finito de alternativas.

Os problemas combinatoriais podem ser divididos em “fáceis” e “difíceis” (IBARAKI, 1988). Normalmente, um problema é dito “fácil” se existe um algoritmo com complexidade de ordem polinomial, ou seja, algoritmo de tempo polinomial, pertencente à classe chamada P (*Polynomial time*). No entanto, a maioria dos problemas combinatoriais são ditos “difíceis” e, por esta razão, pertencentes à classe chamada NP (*Non-deterministic Polynomial time*) que podem ser resolvidos em tempo polinomial, porém por um algoritmo não determinístico (SAHNI e GONZALEZ, 1976).

De acordo com a complexidade, a resolução dos problemas combinatoriais pode ser feita por meio dos chamados métodos exatos, métodos heurísticos ou, se combinados, por métodos híbridos.

Os métodos exatos obtêm soluções ótimas, porém, para o problema de *layout* não são viáveis para mais de 8 facilidades devido ao tempo computacional (MARTINS *et al*, 2003). Os algoritmos *branch and bound* (BAZARAA e ELSHAFEI, 1979) e *cutting plane algorithms* (GOMORY, 1958) são alguns exemplos de métodos exatos.

Os métodos heurísticos obtêm uma solução viável, não necessariamente ótima, mas com maior rapidez que os métodos exatos e, para problemas mais complexos. Os algoritmos mais conhecidos e utilizados são os algoritmos genéticos (HOLLAND, 1975), a busca tabu (GLOVER, 1989a; 1989b) e *simulated annealing* (KIRKPATRICK *et al*, 1983).

Os métodos heurísticos classificam-se em duas grandes classes:

- Algoritmos construtivos: gera a solução final a partir da estaca zero, atribuindo valores a uma ou mais variáveis de decisão ao mesmo tempo.
- Algoritmo de melhoramento: geralmente inicia sua execução com uma solução inicial viável e tenta obter uma solução melhor de forma iterativa. A cada iteração do algoritmo, a qualidade da nova solução é avaliada através do resultado obtido com a função objetivo.

2.2 O problema de *layout* de facilidades

O problema de *layout* pertence à classe NP-*hard*, fácil de entender, mas difícil de resolver, e aparece em muitos campos de aplicações: instalações industriais, instalações comerciais, supermercados, bancos, escritórios, entre outros. Entre os trabalhos pioneiros, destacam-se Koopmans e Beckman (1957), Armour e Buffa (1963), Nugent *et al* (1968) e Vollman e Buffa (1966).

O problema de *layout* de facilidades se caracteriza como um problema de minimização combinatorial. Tem uma função objetivo a ser minimizada, que é definida sobre um conjunto discreto, cujos elementos, são todas as possíveis alocações de instalações. O número de elementos do *layout* cresce fatorialmente com o aumento do número de facilidades *N*, de modo que a busca não pode ser feita de forma exaustiva quando *N* é grande. Também, pelo

fato de tal conjunto ser discreto, impossibilita o uso das noções de continuidade, dificultando nos processos que utilizam o conceito de direção para caminhar no sentido do ótimo (CORTES, 1996).

No caso do projeto de *layout* de setores dentro de um prédio, o número de *layouts* é igual a fatorial de *N*, sendo *N* o número de setores (facilidades) a arranjar. No *layout* de máquinas a complexidade aumenta, pois, além do espaço das máquinas, deve-se considerar o espaço para circulação, o espaço para operação e manutenção dos equipamentos e áreas para futuras expansões (MARQUES, 1993).

O *layout* industrial é a representação espacial dos fatores que concorrem para a produção envolvendo homens, materiais e equipamentos (facilidades), e as suas interações. Resolver o problema de *layout* consiste em encontrar a melhor forma de fazer isso, objetivando, principalmente:

– Facilitar o processo de trabalho, usar eficientemente a força de trabalho e promover o conforto e a segurança do empregado, bem como seu interesse. Consiste em minimizar a movimentação de materiais e pessoas, normalmente pela aproximação de equipamentos e pontos de estocagem. A aproximação de equipamentos também leva uma melhor utilização da mão-de-obra, uma vez que reduz o trabalho improdutivo e permite a diversificação das atividades. Além disso, uma aproximação de equipamentos também aproxima as pessoas, que podem se relacionar melhor no ambiente de trabalho (GAITHER e FRAZIER, 2001);

– Maximizar a flexibilidade e a produtividade. A flexibilidade das operações passa pela facilitação do processo de trabalho, tendo reflexo direto sobre a produtividade. A proximidade de equipamentos e pessoas permite a identificação e redução de problemas de qualidade, aumentando a produtividade do sistema;

– Fazer uso econômico da área. Um bom e racional aproveitamento do espaço pode ter reflexos sobre aspectos estratégicos (expansão e diversificação da produção), uma vez que, se houver falta de espaço, a compra de novos equipamentos será dificultada ou não será efetuada.

Um bom sistema de *layout* é extremamente importante para racionalizar as atividades envolvidas. Isso é igualmente importante para na implementação de sistemas industriais e suas operações diárias (ALVARENGA, 2000).

Os *layouts* podem ser construídos utilizando-se diferentes algoritmos, os quais podem ser divididos em quatro classes (FURTADO e LORENA, 1997a; TAVARES, 2000):

- Algoritmos Construtivos: as facilidades são atribuídas a uma localização, uma de cada vez, até que um *layout* completo seja obtido. Exemplos: ALDEP (SEEHOF e EVANS, 1967), CORELAP (LEE e MOORE, 1967), NLT (VAN CAMP *et al*, 1991), entre outros;

- Algoritmos de Melhoramento: inicia o procedimento a partir de uma solução inicial e, através de permutações de facilidades, busca-se melhorar esta solução inicial. Exemplos: CRAFT (ARMOUR e BUFFA, 1963); COFAD (TOMPKINS e REED Jr., 1976), entre outros;

- Algoritmos Híbridos: combinam algoritmos construtivos com algoritmos de melhoramento. Destacam-se os trabalhos de Elshafei (1977), Scriabin e Vergin (1985), entre outros;

- Algoritmos baseados na Teoria dos Grafos: usam conceitos de grafos planos e de grafos planos máximos. Os trabalhos de Foulds e Robinson (1978) e MIP (*mixed integer program*) (MONTREUIL, 1990; MONTREUIL *et al*, 1993; HERAGU e KUSIAK, 1991) são alguns exemplos.

Independente do algoritmo utilizado para a construção do *layout*, a função objetivo e as restrições, nele inseridas, determinam a qualidade do *layout*.

A função objetivo visa, na maioria dos casos, minimizar o custo entre as facilidades e/ou maximizar a proximidade entre elas, reduzindo a distância. O custo e a distância são as principais variáveis que compõem a função objetivo. O custo, neste caso, refere-se ao fluxo de material entre as facilidades, podendo ser especificado quantitativamente, por um peso previamente atribuído ao fluxo de material, ou qualitativamente, pela adjacência das facilidades, que devem apresentar, no mínimo, dois pontos em comum (MARTINS *et al*, 2003).

A distância entre as facilidades, citada anteriormente, pode ser calculada de várias formas, sendo as seguintes as mais utilizadas (TAVARES, 2000):

- Distância Euclidiana: define o comprimento do segmento da reta que une os centros das instalações. A distância entre as facilidades *i* e *j* é dada por: $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

- Distância Euclidiana Quadrática: esta medida atribui uma maior preponderância aos pares de facilidades que se encontram afastados: $d_{ij} = (x_i - x_j)^2 + (y_i - y_j)^2$.

- Distância Retangular (Distância Retilínea ou, ainda, distância *Manhattan*): verifica a distância entre dois pontos, entre os quais pode haver somente movimento de uma forma retilínea. Devido a sua simplicidade, é a fórmula mais utilizada: $d_{ij} = | x_i - x_j | + | y_i - y_j |$.

Resolver o problema de *layout* consiste em duas etapas gerais: primeiramente, modelar o problema de acordo com as características das facilidades envolvidas e do local no qual elas serão posicionadas e de acordo com o objetivo desejado (função objetivo) e, após, criar, efetivamente, o *layout* através da aplicação de um método heurístico, respeitando-se as restrições impostas.

Dimensões da indústria, área física e formato das facilidades, orientação da facilidade (vertical ou horizontal – variável ou fixa), ocorrência de sobreposição, percentual de variação de área quanto às dimensões, áreas fixas (pilares, escadas, elevadores, etc.) são algumas das restrições que podem ser consideradas na geração de um *layout* de qualidade (MARTINS *et al*, 2003).

2.3 Modelagem do problema de *layout*

O problema de *layout* tem sido modelado utilizando-se: Problema Quadrático de Alocação (KOOPMANS e BECKMAN, 1957), Teoria dos grafos (FOULDS e ROBINSON, 1978), Árvores Binárias (TAM e LI, 1991; FURTADO e LORENA, 1997b; AZADIVAR e WANG, 2000), modelo *Attractor-Repeller* (ANJOS e VANNELLI, 2002), modelo *DISpersion-CONcentration* (DREZNER, 1980; 1987), modelo *Spring-Embedding* (CASTILLO e SIM, 2003), entre outras.

2.3.1 Problema Quadrático de Alocação

O Problema Quadrático de Alocação (PQA) consiste em encontrar uma alocação de custo mínimo (para problemas de minimização) das facilidades aos locais, sendo os custos obtidos pela soma dos produtos distância-fluxo (LOIOLA *et al*, 2004). Dentre os problemas

de otimização combinatória, o PQA é um dos mais difíceis no que diz respeito à complexidade computacional.

No PQA os departamentos precisam ser atribuídos a localizações, respeitando as seguintes restrições: cada localização deve ter um único departamento associado, e cada departamento deve ser atribuído a uma única localização. Um valor positivo é associado a cada par de departamentos, indicando o fluxo entre eles.

O primeiro trabalho sobre PQA foi publicado em 1957, por Koopmans e Beckman, para modelar o problema de alocar e interagir plantas de áreas e formatos iguais. Desde então, têm sido inúmeras as aplicações do problema para representar modelos logísticos, econômicos e de planejamento e aparece em diversas situações práticas, como em projeto de circuitos eletrônicos, em problemas de escalonamento de horário, análise estatística, em planejamentos de hospitais, dentre muitos outros (SALOMÃO, 2005).

Muitos pesquisadores utilizam o Problema Quadrático de Alocação para modelar o problema de *layout* devido à sua importância prática e teórica, dentre eles Elshafei (1977) que o utilizou em planejamentos de hospitais e Dickey e Hopkins (1972), na modelagem de localização de construções em campus universitário.

2.3.2 Teoria dos Grafos

Em 1978, Foulds e Robinson criaram uma nova formulação baseada na teoria de grafos, na qual cada vértice representa uma facilidade e cada aresta representa a possibilidade de atribuição de facilidades adjacentes.

Furtado e Lorena (1997a) mostraram como o problema de *layout* pode ser modelado utilizando-se grafos. Sejam:

$G = (V, E)$ - grafo com conjunto V não vazio de vértices (facilidades) e E conjunto de arestas;

H_{ij} - razão de aproximação, indicando a medida desejável de localizar a facilidade i adjacente à facilidade j (tráfego)*;*

B - conjunto de pares de facilidades que necessitam estar adjacentes em qualquer possível solução;

F - conjunto de pares de facilidades que não necessitam estar adjacentes em qualquer possível solução; e

$E' = \{ \{ i, j\} : a_{ij} = 1, \{i, j\} \in E \},$

$a_{ij} = 1$ se a facilidade *i* é adjacente à facilidade *j*,

$a_{ij} = 0$ caso contrário.

A equação na teoria de grafos é a seguinte:

$$\text{Maximizar} \sum_{i \in E} \sum_{j \in E} H_{ij} a_{ij} , \tag{1}$$

sujeito a $a_{ij} = 1, \{i, j\} \in B,$

$a_{ij} = 0, \{i, j\} \in F,$

$(V, E' \cup B)$ é um grafo planar, ou seja, aquele que pode ser mapeado no plano sem que quaisquer duas arestas se interceccionem.

2.3.3 Árvores Binárias

Furtado e Lorena (1997b) fizeram uso de árvore binária para modelar o problema de *layout*. A árvore é construída de acordo com o fluxo entre as facilidades, tal que as facilidades de maior fluxo fiquem posicionadas próximas umas das outras. Após, a árvore binária é percorrida recursivamente da esquerda para a direita, particionando as facilidades, para gerar o *layout*. O particionamento é realizado de acordo com o valor obtido pelo somatório das áreas das facilidades que estão sendo analisadas (FURTADO e LORENA, 1997b).

Como as folhas da árvore binária constituem as facilidades e a busca é realizada da esquerda para a direita, o primeiro nó esquerdo da subárvore esquerda é analisado e as áreas de todas as suas folhas são somadas para gerar o primeiro particionamento do *layout*. Da mesma forma se existir um nó à esquerda do anterior. Quando não houver mais nós à esquerda, o algoritmo verifica os nós à direita. Pode acontecer, porém, de um nó da direita possuir nó à esquerda, sendo estes analisados primeiramente. Após percorrer todos os nós e folhas da subárvore esquerda, a busca é realizada na subárvore direita à raiz.

2.3.4 Modelo DISpersion-CONcentration

Drezner (1980; 1987) desenvolveu o modelo *DISpersion-CONcentration* (DISCON) para modelagem do problema de *layout*. Para isso, o autor considera cada facilidade como sendo um círculo, a posição da facilidade – que é determinada pelo seu centro denotado pelas coordenadas *x* e *y* –, a distância entre duas facilidades – calculada com a fórmula da distância euclidiana entre os seus centros – e o custo (tráfego) de material manipulado entre duas facilidades (DREZNER, 1980; 1987).

Função objetivo do modelo DISCON:

$$\text{Minimizar} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} d_{ij} \tag{2}$$

A única restrição que este modelo considera é que não haja sobreposição, ou seja,

$$r_i + r_j - d_{ij} \leq 0, \ \forall \ 1 \leq i < j \leq N \tag{3}$$

O autor resolve o modelo DISCON usando um algoritmo de penalidade e uma abordagem em duas fases: na fase *DISpersion* todos os departamentos são distribuídos o mais distante da origem, fornecendo um ponto de início para a próxima fase, a fase *CONcentration*, na qual todos os departamentos estão concentrados e o arranjo resultante é a solução final.

Este método não leva em consideração as dimensões das facilidades, não exige que todos os departamentos sejam colocados dentro da grande área (área da empresa), nem mesmo a dimensão desta.

2.3.5 Modelo Attractor-Repeller

O modelo *Attractor-Repeller* – AR (ANJOS e VANNELLI, 2002) considera cada facilidade como um círculo, cuja posição é determinada pelo seu centro, denotado pelas coordenadas *x* e *y*. A função objetivo mostrada na Equação 4 visa minimizar a relação custo e distância entre as facilidades, atraindo-as.

$$\text{Minimizar F} = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} d_{ij} \tag{4}$$

Na equação, N representa o número total de facilidades, c_{ij}, o custo entre as facilidades i e j e d_{ij}, a distância euclidiana entre elas, dada por:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{5}$$

Este modelo utiliza o conceito de distância alvo, t_{ij}, que age como uma força repulsiva entre as facilidades, sendo esta calculada da seguinte forma,

$$t_{ij} = \alpha (r_i + r_j), \forall\ 1 \leq i < j \leq N, \tag{6}$$

onde r_i se refere ao raio da facilidade i e r_j, ao raio da facilidade j. O parâmetro α controla a sobreposição entre as facilidades e, de acordo com Castillo e Sim (2003), normalmente utiliza-se $\alpha > 0$, pois $\alpha < 1$ permite alguma sobreposição e $\alpha = 1$ não há sobreposição, mas separa demais as facilidades. Sempre que houver sobreposição entre duas facilidades, a função objetivo é penalizada com a introdução do termo,

$$f\left(\frac{d_{ij}}{t_{ij}}\right), \text{ onde f}(z) = \frac{1}{z} - 1,\ z > 0, \tag{7}$$

sendo representada da seguinte forma, onde P é uma constante maior que zero,

$$\text{Minimizar F} = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} d_{ij} + P \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} f\left(\frac{d_{ij}}{t_{ij}}\right). \tag{8}$$

A Figura 1 mostra como as forças de atração e repulsão agem sobre as partículas.

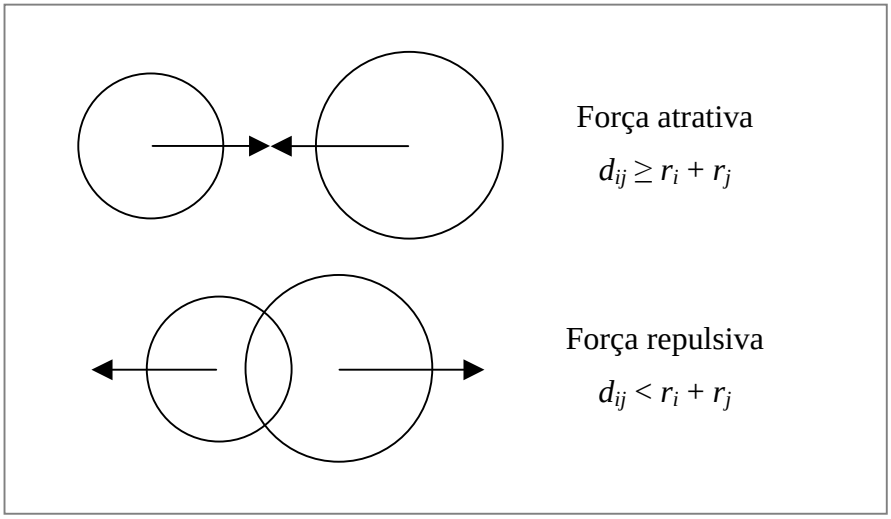


Figura 1 – Representação das forças atrativa e repulsiva entre as facilidades, no modelo *Attractor-repeller*

Fonte: Castillo e Sim, 2003

Toda vez que a soma dos raios de duas facilidades forem menor ou igual à distância euclidiana entre elas ocorre uma força atrativa entre elas. Do contrário, ou seja, quando a soma dos raios de duas facilidades forem maior que a distância euclidiana entre elas significa uma sobreposição, então, uma força repulsiva age sobre elas.

O modelo AR considera as seguintes restrições:

$$r_i + r_j - d_{ij} \leq 0, \forall\ 1 \leq i < j \leq N, \quad (9)$$

$$x_i + r_i - \tfrac{1}{2}wi \leq 0,\ i = 1, ..., N, \quad (10)$$

$$r_i - x_i - \tfrac{1}{2}wi \leq 0,\ \ i = 1, ..., N, \quad (11)$$

$$y_i + r_i - \tfrac{1}{2}he \leq 0,\ \mathrm{i} = 1, ..., N, \quad (12)$$

$$r_i - y_i - \tfrac{1}{2}he \leq 0,\ i = 1, ..., N, \quad (13)$$

nas quais N representa o número total de facilidades, r_i e r_j representam os raios das facilidades i e j, x_i e y_i representam as coordenadas da facilidade i, wi e hi, a largura e a altura da área do *layout*, respectivamente. A Equação 9 é usada para evitar a sobreposição das facilidades e as Equações 10 a 13, para impedir que as facilidades saiam da área total do *layout* (CASTILLO e SIM, 2003).

Anjos e Vannelli (2002) utilizaram o modelo AR para tentar obter melhores *layouts*. Inicialmente utilizaram $\alpha = 1$, mas constataram que as facilidades ficavam muito separadas umas das outras. Então, utilizaram $\alpha = 0.005$, o que resolveu o problema, no entanto, gerou

alguma sobreposição. Os resultados apresentados pelos autores mostram que o valor escolhido, aleatoriamente, para α possui influência significativa na solução final do *layout*. Outros trabalhos similares foram desenvolvidos por Heragu e Kusiak (1991) e Van Camp *et al* (1991).

3 HEURÍSTICAS E O PROBLEMA DE *LAYOUT*

As heurísticas têm sido amplamente estudadas nas últimas décadas e, na maioria dos casos, têm-se mostrado superiores aos métodos exatos, mesmo obtendo como resultado soluções aproximadas ao ótimo.

Algumas heurísticas caracterizam-se por serem técnicas inspiradas em fenômenos da natureza, fáceis de implementar e de grande eficiência. São técnicas classificadas como estocásticas, de caráter aleatório, ou seja, a cada execução do algoritmo, o resultado obtido é diferente do anterior. As mais conhecidas são: Algoritmos Genéticos (HOLLAND, 1975), *Simulated Annealing* (KIRKPATRICK *et al*, 1983), Busca Tabu (GLOVER, 1989a; 1989b), Colônia de Formigas (DORIGO, 1992) e Enxame de Partículas (KENNEDY e EBERHART, 1995), que será utilizada neste trabalho.

3.1 Algoritmos Genéticos e o Problema de *Layout*

Introduzido por John Holland (1975), algoritmos genéticos são algoritmos de busca baseados na genética humana e, basicamente, estruturado da seguinte forma: uma população inicial é gerada, na qual cada indivíduo (cromossomo), representado por uma string de 0s e 1s (genes), é avaliado pela função objetivo que, no processo de seleção, descarta indivíduos com baixa qualidade. A população, então, sofre recombinação através de duas operações principais: o *crossover* – que gera novos indivíduos através da combinação de outros dois escolhidos aleatoriamente – e a mutação – que modifica um dos genes de um indivíduo.

Em 1992, Tam utilizou algoritmos genéticos representado pela estrutura de árvore binária na tentativa de resolver o problema de *layout*. Os nós da árvore representam a identificação das facilidades (operandos) ou símbolos de corte (operadores). Cada nó interno representa o tipo de corte da facilidade, representado por uma letra, e cada folha da árvore binária representa a identificação da facilidade, como mostra a Figura 2.

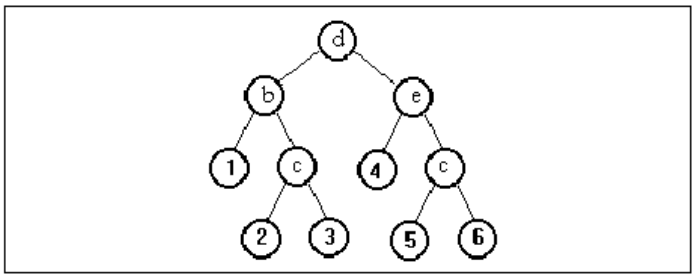


Figura 2 – Representação de uma árvore binária utilizada para resolver o problema de *layout* com o método algoritmo genético
Fonte: Furtado e Lorena, 1997b.

A seleção dos parâmetros: tamanho da população, taxa para crossover e mutação foram baseados nos estudos de Grefenstette (1986) e Schaffer *et al* (1989). Quatro *layouts* com 12, 15, 20 e 30 facilidades foram analisados (NUGENT *et al*, 1968), cuja solução inicial foi gerada aleatoriamente, determinando operadores de corte para 30 nós da árvore. A performance do algoritmo genético foi comparada com o método *hill-climbing*, que busca uma vizinhança *G*, onde *G* é um conjunto de seqüência de operadores gerados da troca de um operador. Para *layout* de 30 facilidades os algoritmos genéticos melhoraram o resultado mínimo em 10,5% e o resultado médio em 13% (MAVRIDOU e PARDALOS, 1997).

3.2 Simulated Annealing e o Problema de *Layout*

Simulated Annealing é um algoritmo de busca local que explora a analogia entre os problemas de otimização combinatória e os da mecânica estatística (KIRKPATRICK *et al*, 1983). A técnica se refere ao processo de resfriar um material diminuindo a temperatura até atingir o equilíbrio. Para tal, leva em consideração: a temperatura – que controla a probabilidade de aceitação de trocas –, o equilíbrio – condição na qual uma melhora na solução usando trocas adicionais é improvável de ocorrer – e a ordem de recozimento – que determina quando e o quanto a temperatura será reduzida.

Valores muito altos de temperatura são importantes para manter a diversidade do espaço de solução, mas podem dificultar o algoritmo a convergir para uma solução boa. Valores muito baixos, ao contrário, forçam a convergência rápida do algoritmo, mas podem levá-lo a estacionar em ótimos locais. Portanto, o equilíbrio desses fatores pode ser decisivo para o sucesso do algoritmo.

Heragu e Alfa (1992) apresentam uma análise experimental do uso do *simulated annealing* para o problema de *layout*. O principal passo do algoritmo consiste em alterar a posição de duas facilidades, aplicar a função objetivo e, se o valor for menor que o encontrado anteriormente, realizar a troca. Isso é repetido até 100*N* iterações ou até que o número de novas soluções aceitas for igual a 10*N*, onde *N* é o número de facilidades no problema de *layout*. Após, o algoritmo decrementa o valor da temperatura baseado na razão de resfriamento e repete o passo principal.

Burkard e Rendl (1984) modelaram o problema de *layout* utilizando problema quadrático de alocação e o método *simulated annealing*. Eles obtiveram bons resultados computacionais e superaram entre 1 e 2% das melhores soluções conhecidas na época.

O estudo de Alvarenga (2000) mostra como o problema de *layout* pode ser resolvido utilizando *simulated annealing*. O algoritmo inicia com uma solução inicial aleatória e com um valor relativamente alto para a temperatura inicial. O algoritmo é composto de dois laços de iteração. O laço interno é repetido para fixar a temperatura enquanto que, no laço externo, a temperatura é reduzida até que a temperatura final seja atingida.

Para iterações no laço interno, a seguinte estratégia heurística é usada para aproximar o estado de equilíbrio: dada uma solução corrente, representada por uma lista de números de facilidades, geram-se as soluções vizinhas, permutações das facilidades *i* e *j* na lista como segue: sendo *N* o número de facilidades, para cada *i* de 1 até (*N*-1), escolhe-se *j* de *i*+1 para *N*. *N*o final deste laço, a solução obtida é comparada com a atual melhor solução e esta é atualizada se for obtida solução melhor.

Supondo um conjunto de 4 facilidades e que nesta iteração a solução corrente é 3, 4, 1, 2. Com esta solução, são geradas todas as possíveis soluções vizinhas utilizando a heurística proposta:

3, 4, 2, 1 1, 4, 3, 2 3, 1, 4, 2

2, 4, 1, 3 3, 2, 1, 4 4, 3, 1, 2

Quando todas as soluções vizinhas geradas tiverem sido avaliadas pelo algoritmo *simulated annealing*, o equilíbrio é alcançado. A Tabela 1 mostra os resultados obtidos pelo algoritmo e faz uma comparação com o melhor resultado encontrado na literatura.

Tabela 1 – Resultados da aplicação do método *simulated annealing* ao problema de *layout*

Número de departamentos	**Melhor solução encontrada na literatura**	**Solução com *simulated annealing***	
		Valor da Função objetivo	**Tempo(s)**
4	225.000	225.000	0.170
5	1.165	1.100	0.330
6	2.085	1.990	0.710
7	5.420	4.730	1.270
8	2324.500	2324.500	2.080
10	2781.500	2781.500	5.490
11	6933.500	6933.500	7.960
20	15602.000	15549.000	85.300
30	45111.000	44965.000	463.740

Fonte: Alvarenga, 2000

Observa-se que, na maioria dos casos, houve uma melhora da solução final e, nenhum resultado apresentou qualidade inferior àqueles encontrados na literatura.

Koakutsu e Hirata (1992) propuseram um algoritmo chamado *Genetic Simulated Annealing* (GSA) para o problema de *layout* de circuitos elétricos. O problema consiste em alocar um determinado conjunto de módulos retangulares num plano com o objetivo de minimizar a área destinada a todos os módulos (plano) e o comprimento da ligação entre os módulos que devem ser conectados no circuito. As características principais do algoritmo são: otimização estocástica, caminho de busca múltipla, seleção de trajetos de busca e o uso do operador genético *crossover* para gerar uma nova solução (MAVRIDOU e PARDALOS, 1997).

3.3 Busca Tabu e o Problema de *Layout*

A Busca Tabu (GLOVER, 1989a; 1989b) é uma metaheurística cujo diferencial está na lista tabu que guarda as trocas classificadas como tabu, que são impedidas, por um certo tempo, de serem realizadas. O algoritmo funciona basicamente assim: partindo-se de uma

solução inicial, a busca tabu explora a vizinhança e, através de movimentos que não sejam tabus, ou seja, impedidos de serem realizados, gera-se uma nova solução que é avaliada através da função objetivo. Se a solução encontrada for melhor que a anterior, acontece a atualização da solução corrente. A lista tabu é atualizada acrescentando-se a troca efetuada e decrementando o tempo tabu (tempo em que uma solução permanece na lista tabu) das trocas já contidas nela.

Alvarenga (2000) utilizou a busca tabu na tentativa de resolver o problema de *layout*. A heurística proposta é similar ao algoritmo *simulated annealing*: para cada *i* variando de 1 até (*N*-1) e *j* variando de *i*+1 até *N*, uma troca entre as facilidades *i* e *j* é executada. Os resultados foram considerados satisfatórios comparados com o que havia na literatura até o momento da pesquisa, conforme mostra a Tabela 2.

Tabela 2 – Resultado da aplicação do método busca tabu ao problema de *layout*

Número de departamentos	**Melhor solução encontrada na literatura**	**Solução com busca tabu**	
		Valor da Função objetivo	**Tempo(s)**
4	225.000	225.000	0.050
5	1.165	1.100	0.050
6	2.085	1.990	0.050
7	5.420	4.730	0.110
8	2324.500	2324.500	0.220
10	2781.500	2781.500	0.490
11	6933.500	6933.500	0.770
20	15602.000	15549.000	9.940
30	45111.000	44965.000	34.490

Fonte: Alvarenga, 2000

Observa-se que os valores encontrados com o algoritmo busca tabu foram os mesmos encontrados utilizando-se o algoritmo *simulated annealing*, apresentados na Tabela 1. No entanto, o tempo de processamento do primeiro algoritmo é bastante inferior.

Martins *et al* (2003) construíram uma ferramenta computacional chamada “Ambiente Visual para Otimização de *Layout* Industrial” (AVOLI), que faz uso da heurística busca tabu para gerar *layouts* de qualidade.

Inicialmente foi gerada uma lista de alocação contendo a solução inicial da seguinte forma: o maior departamento é alocado e, após, a alocação é feita de acordo com o grau de relacionamento entre os departamentos, ou seja, o próximo a ser alocado será aquele que

possuir maior relacionamento com aquele inserido anteriormente. Se isso não ocorrer, então o próximo departamento a ser alocado é o maior deles. Após, a alocação segue de acordo com o grau de relacionamento.

Após gerada a lista de alocação, é executado o algoritmo de alocação, responsável por posicionar os elementos da lista no quadro externo. A alocação é feita da esquerda para direita e de cima para baixo e o algoritmo busca tabu é, então, executado. Martins *et al* (2003) realizaram os testes utilizando oito problemas-teste (5, 6, 7, 8, 12, 15, 20 e 30 departamentos) propostos por Nugent *et al* (1968) e, baseado neles, realizaram quatro estudos de casos.

No primeiro estudo de caso, Martins *et al* (2003) utilizou tempo tabu fixo igual a 5 para os quatro primeiros problemas e 8 para os demais. Utilizou também estrutura de vizinhança e pseudo-troca simples de um determinado tamanho, e o custo entre cada par de departamentos foi considerado unitário. O resultado foi comparado com os resultados obtidos por outros quatro autores e não apresentou melhoras, como pode ser visto na Tabela 3, apresentada pelos autores.

Tabela 3 – Comparativo entre os resultados obtidos pelo AVOLI e por outros pesquisadores: primeiro estudo de caso

Total de departamentos	**Total de iterações**	**Nugent (1968)**	**Tate (1995)**	**Surech (1995)**	**Chiang (2001)**	**AVOLI**
5	100	50	50	50	50	50
6	100	86	86	86	86	86
7	100	148	148	148	144	148
8	100	214	214	214	212	214
12	300	578	578	578	578	578
15	200	1.150	1.150	1.150	1.110	1.150
20	500	2.570	2.598	2.570	2.564	2.570
30	500	6.124	6.184	6.168	6.094	6.128

Fonte: Martins *et al*, 2003

No segundo estudo de caso, o autor utilizou os mesmos problemas-teste e as dimensões dos departamentos utilizadas por Tam e Li (1991) e adotou o máximo de 1000 iterações. Para fins de comparação, Martins *et al* (2003) realizaram os testes utilizando métodos exatos para os problemas com até 8 departamentos e a heurística busca tabu para todas as instâncias. De acordo com os resultados apresentados na Tabela 4, a heurística obteve os mesmos resultados obtidos pelos métodos exatos, comprovando, assim, sua eficácia. Para problemas envolvendo

mais de 12 departamentos não foi possível a comparação pois métodos exatos não são capazes de resolvê-los. O percentual indicado na tabela é referente à melhora da solução inicial.

Tabela 4 – Resultados obtidos com a aplicação da heurística busca tabu ao problema de *layout*: segundo estudo de caso

Total de departamentos	Quadro externo	Total de iterações	Solução inicial	Solução final		Melhoria (%)
				Exata	Heurística	
5	14 x 09	250	272	232	232	14,71
6	13 x 11	250	456	376	376	17,54
7	16 x 10	250	1.030	676	676	24,72
8	18 x 10	250	1.210	974	974	19,5
12	19 x 13	1.000	3.229	-	2.347	27,31
15	20 x 15	500	7.104	-	4.853	31,69
20	21 x 15	1.000	13.779	-	9.722	29,44
30	25 x 16	1.000	34.274	-	23.312	31,98

Fonte: Martins *et al*, 2003

Para o terceiro estudo de caso, foram utilizados os problemas com 8, 12, 15, 20 e 30 departamentos de dimensões diferentes e com algumas restrições quanto à orientação dos departamentos e proximidade entre eles. Foram consideradas áreas ocupadas e departamentos fixos. Conforme os resultados obtidos, mostrados na Tabela 5, houve uma melhora significativa em relação à solução inicial.

Tabela 5 – Resultados obtidos com a aplicação da heurística busca tabu ao problema de *layout*: terceiro estudo de caso

Total de departamentos	Quadro externo	Total de iterações	Solução inicial	Solução final	Melhoria (%)
8	10 x 17	1.000	1.172	898	23,38
12	18 x 13	1.000	4.174	2.744	34,26
15	20 x 14	1.000	6.870	5.306	22,77
20	21 x 15	1.000	17.502	10.338	28,37
30	25 x 16	1.000	28.660	21.888	22,58

Fonte: Martins *et al*, 2003

No último estudo de caso, Martins *et al* (2003) utilizaram problemas com 40 departamentos, com quadro externo de 60 x 45, e com 50 departamentos, com quadro externo de 90 x 60. A Tabela 6 mostra os resultados obtidos com a aplicação da heurística busca tabu ao problema de *layout* e, para este estudo, verifica-se uma considerável melhora em relação à solução inicial.

Tabela 6 – Resultados obtidos com a aplicação da heurística busca tabu ao problema de *layout*: quarto estudo de caso

Total de departamentos	**Quadro externo**	**Total de iterações**	**Solução inicial**	**Solução final**	**Melhoria (%)**
40	60 x 45	500	111.027	78.755	29,07
55	90 x 60	500	187.770	130.930	30,27

Fonte: Martins *et al*, 2003

Os autores constataram, a partir de todos os resultados obtidos, que a escolha adequada das dimensões do quadro externo é imprescindível para a melhoria dos resultados.

4 ENXAME DE PARTÍCULAS

4.1 Histórico

O método de otimização por enxame de partículas (*Particle Swarm Optimization* – PSO) foi desenvolvido por Kennedy e Eberhart, em 1995, a partir do trabalho do biólogo Frank Heppner, que analisou o comportamento de um grupo de pássaros a procura de alimento ou de um lugar para construir o ninho, conforme ilustrado na Figura 3.

O método faz uma simulação do “comportamento social” dos pássaros, ou seja, quando um pássaro encontra o alimento, por exemplo, todos os demais passam a encontrá-lo também, mais rapidamente. O que acontece é um aprendizado por parte do bando no momento que um dos pássaros adquire determinado conhecimento (GOMES, 2004). Prado e Saramago (2005) chamam esse processo de “inteligência social”.

Figura 3 – Representação do método Enxame de Partículas

Enxame de partículas é similar aos métodos de computação evolucionária em que, uma população (enxame), formada por indivíduos (partículas) vasculham o espaço de busca à procura de uma solução apropriada para um determinado problema. Entretanto, na otimização por enxame de partículas, cada indivíduo tem uma velocidade, responsável pela exploração do espaço (evolução) e uma memória, para guardar a melhor posição já visitada (EBERHART *et al*, 1996). Além disso, o algoritmo considera também, a melhor posição encontrada pela população.

Segundo Parsopoulos e Vrahatis (2002), cada partícula é tratada como um ponto dentro do espaço de busca, que ajusta seu próprio "vôo" de acordo com sua própria experiência, bem como a experiência do "vôo" de outras partículas.

Enxame de Partículas é um algoritmo estocástico de conceito simples, fácil implementação, robustez para controlar parâmetros e eficiência computacional durante o processo de otimização (PARK *et al*, 2005).

4.1.1 O algoritmo de otimização por enxame de partículas

Considerando um espaço bidimensional, cada partícula possui uma posição no espaço de soluções (x e y) e uma velocidade que permite a ela percorrer esse espaço. A cada iteração, e para cada partícula, a velocidade ($\boldsymbol{v_i}$) é atualizada, conforme Equação 14, de acordo com a velocidade anterior (v_i) somada à parte cognitiva da fórmula (c_1 * rand() * ($pbest_i - x_i$)), que representa o conhecimento, e à parte social (c_2 * rand() * ($gbest - x_i$)), que representa a colaboração entre as partículas. A nova posição da partícula é determinada pela soma da sua posição atual e a nova velocidade (EBERHART *et al*, 1996), de acordo com a Equação 15.

$$\boldsymbol{v_i}^{\boldsymbol{it+1}} = v_i^{\,it} + (c_1^{\,it} * \text{rand}()^{it} * (pbest_i^{\,it} - x_i^{\,it}) + (c_2^{\,it} * \text{rand}()^{it} * (gbest^{\,it} - x_i^{\,it}), \quad (14)$$

$$\boldsymbol{x_i}^{\,it+1} = x_i^{\,it} + \boldsymbol{v_i}^{\,\boldsymbol{it+1}} \quad (15)$$

onde:

v_i – velocidade atual da partícula i

c_1, c_2 – parâmetros de confiança

rand() – função aleatória

$pbest_i$ – melhor posição que a partícula i já obteve durante a busca

gbest – melhor posição encontrada pelas partículas no enxame
x_i – posição atual da partícula *i*
it – iteração atual

As equações 14 e 15 definem a primeira versão do algoritmo de otimização por enxame de partículas. O algoritmo armazena as velocidades num vetor $V_i = (v_1, v_2, ..., v_n)$, assim como as posições das partículas $X_i = (x_1, x_2, ..., x_n)$ e as melhores posições já encontradas pelas partículas $pbest_i = (pbest_1, pbest_2, ..., pbest_n)$.

4.1.1.1 Parâmetros de confiança

Propostos por Shi e Eberhart (1998), os parâmetros de confiança indicam quanto uma partícula confia em si (c_1) e no enxame (c_2). Os valores dessas variáveis dependem muito do problema em questão e são previamente conhecidos, podendo ser fixos ou variar a cada iteração do algoritmo.

Em seus estudos, os autores utilizaram $c_1 = c_2 = 2$. No entanto, Kennedy (1998) constatou que $c_1 = c_2 = 0{,}5$ produz resultados melhores.

Ratnaweera *et al* (2004) propuseram as Equações 16 e 17 para variar os parâmetros de confiança a cada iteração do algoritmo:

$$c_1^{it} = \left(c_{1\,fin} - c_{1ini}\right)\frac{it}{R} + c_{1ini} \tag{16}$$

$$c_2^{it} = \left(c_{2\,fin} - c_{2ini}\right)\frac{it}{R} + c2_{ini} \tag{17}$$

onde:
c_{1ini} = valor inicial para o parâmetro de confiança cognitivo
c_{1fin} = valor final para o parâmetro de confiança cognitivo
c_{2ini} = valor inicial para o parâmetro de confiança social
c_{2fin} = valor final para o parâmetro de confiança social
R = Número de iterações
it = iteração atual

4.1.1.2 Função aleatória

A função aleatória rand() é responsável por manter a diversidade do enxame e gera números aleatórios entre 0 e 1.

4.1.1.3 Aceleração por distância

A equação que atualiza a velocidade da partícula é composta por dois termos de aceleração por distância: ($pbest_i - x_i$) e ($gbest - x_i$), no qual o primeiro representa a distância entre a melhor posição já encontrada pela partícula *i* e a sua posição atual, e o segundo termo representa a distância entre a melhor posição encontrada pelo enxame e a posição atual da partícula *i*.

4.1.1.4 Implementação do algoritmo original de otimização por enxame de partículas

O processo original para implementação da versão global do algoritmo de otimização por enxame de partículas está descrito a seguir (EBERHART *et al*, 1996):

1. Inicializar a população de partículas com posições e velocidades aleatórias no espaço *n* dimensional;
2. Para cada partícula:
   Calcular a função objetivo;
   Comparar o valor obtido da partícula *i* com *pbest*. Se o valor for melhor, atualizar *pbest* com o novo valor;
   Comparar o valor obtido com o melhor valor global *gbest*. Se for melhor, atualizar *gbest* com o novo valor;
   Alterar a velocidade e a posição da partícula;
3. Repetir os passos anteriores até que o critério de parado seja satisfeito, sendo este, normalmente definido pelo número de iterações.

Na Figura 4 estão representadas duas partículas, 1 e 2, no espaço bidimensional (*x*, *y*). Cada partícula possui uma posição no espaço, determinada pelas coordenadas, (x_1, y_1) e (x_2, y_2), uma velocidade, v_1 e v_2, e a melhor posição já encontrada, P_1 e P_2, respectivamente.

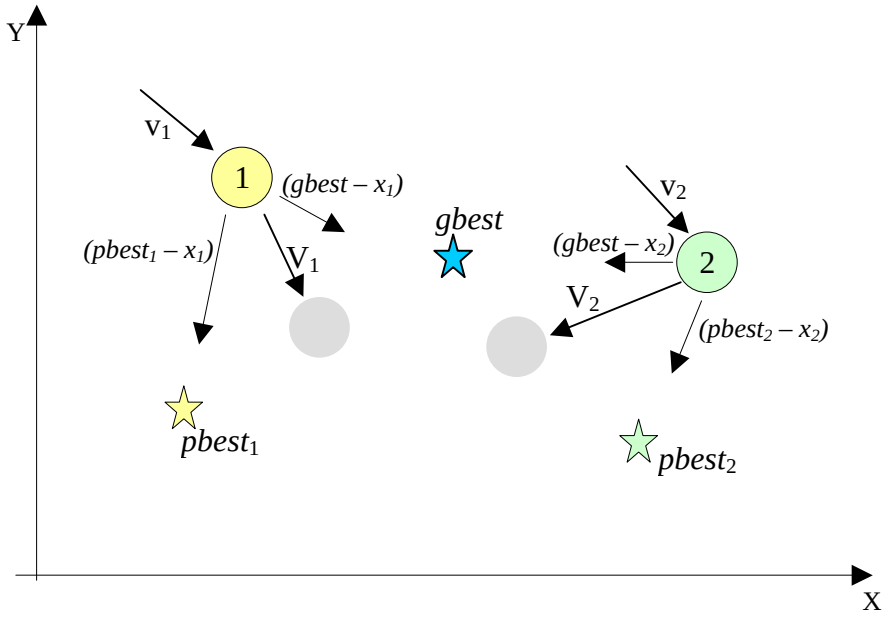


Figura 4 – Comportamento das partículas no espaço de busca bidimensional conforme processo original de otimização por enxame de partículas
Fonte: Prado e Saramago, 2005

onde:

$pbest_i$ – melhor posição da partícula

gbest – melhor posição de todas as partículas no enxame

$(pbest_i - x_i)$ – distância entre a melhor posição da partícula e sua posição atual.

$(gbest - x_i)$ – distância entre a melhor posição de todas as partículas no enxame e sua posição atual

◯ – posição atual

● – próxima posição

A posição na qual a partícula se encontra foi calculada mediante as equações 14 e 15 e, para o cálculo da nova posição, indicada pelo círculo sombreado, as mesmas equações são utilizadas, levando-se em consideração a velocidade anterior da partícula, *v*, a sua posição atual (x, y), a melhor posição que essa partícula já ocupou até o momento, *pbest* e a melhor posição de todas as partículas no enxame, *gbest*, como mostra a Figura 4.

4.1.2 Variantes do algoritmo de otimização por enxame de partículas

Diversos autores propuseram alterações no método original. A seguir, alguns exemplos mais relevantes de aperfeiçoamentos do método original.

4.1.2.1 Limitador da velocidade

Quando criada a primeira versão do algoritmo de otimização por enxame de partículas verificou-se a necessidade de se controlar a velocidade da partícula, já que um valor muito alto pode fazer com que a partícula ultrapasse uma posição ótima e um valor muito baixo pode ser insuficiente para alcançar o ótimo. Foi estabelecido, então, o parâmetro v_{max} como limite máximo da velocidade.

Segundo Kennedy *et al* (2001), o valor de v_{max} é especificado pelo usuário de acordo com o problema, normalmente entre [-4, 4], ou seja, se a nova velocidade ($\boldsymbol{v_i}$) for superior a 4, a ela é atribuído valor 4. Caso a velocidade encontrada for inferior a -4, seu valor passará a ser -4.

No entanto, o limitador de velocidade, apesar de importante, resulta numa baixa eficiência do algoritmo de otimização por enxame de partículas se comparada a outros métodos de Computação Evolucionária (ANGELINE, 1998). E ainda, apesar de localizar a área do ótimo muito mais rápido que os demais métodos, uma vez na região próxima do ótimo, o algoritmo de otimização por enxame de partículas pode não continuar ajustando sua velocidade na tentativa de encontrar uma solução melhor. Para resolver estas questões, foi indicado a utilização de dois parâmetros, utilizados separadamente, junto à equação da velocidade: o componente inercial e o fator de constrição (SHI e EBERHART, 1998).

4.1.2.2 Componente inercial

A maior dificuldade do método Enxame de Partículas é convergir para o ótimo nas iterações finais. Shi e Eberhart (1998) propuseram outra alternativa para melhorar a performance do método Enxame de Partículas que consiste na introdução do componente inercial na equação que atualiza a nova velocidade da partícula:

$$\boldsymbol{v_i}^{\boldsymbol{it+1}} = w^{it} * v_i^{\boldsymbol{it}} + c_1{}^{it} * \text{rand}()^{\boldsymbol{it}} * (pbest_i{}^{\boldsymbol{it}} - x_i{}^{\boldsymbol{it}}) + c_2{}^{it} * \text{rand}()^{\boldsymbol{it}} * (gbest^{\boldsymbol{it}} - x_i{}^{\boldsymbol{it}}) \quad (18)$$

Em 2000, Eberhart e Shi, propuseram que o valor de *w* variasse a cada iteração do algoritmo e, para isso, criaram a seguinte equação:

$$w^{it} = (w_{ini} - w_{fin})\frac{(R - it)}{R} + w_{fin} \quad (19)$$

onde:

w_{ini} = valor inicial para o coeficiente de inércia

w_{fin} = valor final para o coeficiente de inércia

Para e Shi e Eberhart (1998), o ideal é que *w* inicie com um valor alto e vá sendo decrementado a cada iteração para dar equilíbrio entre exploração global e local: um valor alto para *w* gera um comportamento mais global e um valor baixo, um comportamento mais local das partículas. Com isso é possível encontrar soluções mais refinadas, suficientemente ótimas, em um menor número de iterações. Em função disso, os autores propuseram a variação linear do coeficiente entre 0,9 e 0,4. Parsopoulos e Vrahatis (2002) consideram uma boa escolha um valor inicial em torno de 1.2 e gradual declínio para 0.

Chatterjee e Siarry (2006) apresentaram uma proposta de variação não-linear para o componente inercial, facilitando a convergência das partículas nas iterações finais, conforme a seguinte equação:

$$w^{it} = \left\{\frac{(R - it)^{nl}}{R^{nl}}\right\} * (w_{ini} - w_{fin}) + w_{fin} \quad (20)$$

Onde *nl* representa um coeficiente de não linearidade. Os valores inicial e final do componente inercial são definidos previamente e o comportamento dele, durante as iterações, dependerá do valor de *nl*.

Prado e Saramago (2005) utilizaram $c_1 = c_2 = 2$ e $w_0 = 0.729$ para testar o método enxame de partículas e, a cada iteração, fizeram uso da Equação 21 para atualizar a inércia:

$$w_{new} = f_w \, w_{old} \tag{21}$$

O fator de redução, f_w é uma constante entre 0 e 1.

A seleção adequada do valor de *w* dá equilíbrio entre exploração global e local, e resulta em menos iterações para encontrar uma solução suficientemente ótima.

4.1.2.3 Fator de constrição

O fator de constrição controla a magnitude das velocidades, desempenhando função semelhante ao parâmetro v_{max} (Clerc e Kennedy, 2002).

$$\boldsymbol{v_i}^{\boldsymbol{it+1}} = k^{it} \left[v_i^{it} + c_1^{it} * \text{rand}()^{it} * (pbest_i^{it} - x_i^{it}) + c_2^{it} * \text{rand}()^{it} * (gbest^{it} - x_i^{it}) \right] \tag{22}$$

sendo,

$$k^{it} = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|} \tag{23}$$

onde: $\varphi = c_1 + c_2$, com $\varphi > 4$.

4.1.3 Centróide do enxame

O centróide do enxame (*C*) pode ser acrescentado à equação da velocidade para fazer com que as partículas concentrem-se próximas ao centro do enxame, evitando a dispersão das mesmas. Esse modelo foi desenvolvido por Albrecht (2004) e designado como “Atração Social”, pois se refere à influência do grupo sobre a partícula.

Os autores provaram uma maior abrangência na exploração do espaço. O centróide do enxame é calculado da seguinte forma:

$$C = \frac{\sum_{i=1}^{N} x_i}{N} \tag{24}$$

4.1.3.1 Parâmetros de confiança c_3

Ratnaweera *et al* (2004) propuseram a utilização de mais um parâmetro de confiança, c3, em função do centróide, conforme mostra a Equação 25.

$$c_3{}^{it} = \left(c_{3\,fin} - c_{3ini}\right)\frac{it}{R} + c_{3ini} \tag{25}$$

Onde:
c_{3ini} = valor inicial para o parâmetro de atração da partícula pelo grupo
c_{3fin} = valor final para o parâmetro de atração da partícula pelo grupo

Os valores iniciais e finais dos parâmetros de confiança dependem muito do problema. Porém, deve-se ter cuidado para que φ seja maior que 4, quando utilizado o fator de constrição.

Assim, o termo proposto que deverá ser adicionado à equação da velocidade é c_3 * rand() * (C - x_i).

4.2 Enxame de Partículas: versão local

Eberhart e Shi (2001) agregaram o conceito de vizinhança ao método enxame de partículas a fim de aumentar a capacidade de exploração do espaço. Dessa forma, a partícula que exerce influência social passa a ser a mais bem sucedida entre seus vizinhos, o que gerou uma versão local para o método.

Encontra-se na literatura estudos sobre a versão local do algoritmo de otimização por enxame de partículas, na qual o enxame é dividido em vizinhanças, e cada vizinhança age

independente umas das outras. Neste caso, as partículas têm informação somente da sua melhor posição e melhor posição dos seus vizinhos e movem-se para pontos definidos por *pbest*, já conhecido, e *lbest*, que é a partícula com melhor avaliação dentre as partículas vizinhas. *Lbest* substitui o parâmetro *gbest*, utilizado na versão global do algoritmo (EBERHART e SHI, 2001).

Por exemplo, se o tamanho da vizinhança é definido como 2, a partícula *i* compara o valor encontrado com a função objetivo das partículas *i*-1 e *i*+1. As primeiras experiências utilizaram como tamanho da vizinhança 15% do tamanho da população. Por exemplo, para uma população de 40 partículas, considerou-se uma vizinhança de 6 partículas. Porém, este trabalho não se aprofunda nesta variação do algoritmo de otimização por enxame de partículas, o que poderá vir a ser objeto de estudos futuros.

4.3 Aplicabilidade do método

A otimização por enxame de partículas tem sido aplicada em diversas áreas, como por exemplo: análise de tremor humano, incluindo Parkinson, que utiliza o método enxame de partículas junto com redes neurais (EBERHART e HU, 1999); remoção de metal em ambientes industriais (TANDON, 2000); projeto para estabilizar sistemas potentes – *power system stabilizer design* (ABINO, 2002); potência reativa e controle de voltagem (YOSHIDA *et al*, 2000); identificação de bordas em segurança dinâmica – *dynamic security border identification* (KASSABALIDIS *et al*, 2002); engenharia nuclear (GUIMARAES, 2005); minimização da largura de banda – *bandwidth minimization problem* (LIM, LIN e XIAO, 2003).

5 IMPLEMENTAÇÃO DO MÉTODO ENXAME DE PARTÍCULAS PARA O PROBLEMA DE *LAYOUT* E RESULTADOS – ABORDAGEM I

Na concepção original do algoritmo enxame de partículas, as partículas exploram o espaço de busca, guiadas pela sua própria experiência e pela partícula que atingiu a melhor posição no enxame como um todo, ou seja, aquela que está mais próxima da solução do problema.

Tratando-se do problema de *layout*, objeto de estudo desta dissertação, cada facilidade deve tentar se aproximar de outras facilidades que possuem maior custo em relação a ela. Fazendo uso do método enxame de partículas para este problema, além de cada partícula ser guiada pela sua própria experiência, ela também é guiada pela sua posição quando se obteve o melhor *layout*. Além dessa alteração, foram necessárias outras que serão apresentadas ao longo deste capítulo, para adequar o método ao problema.

5.1 O algoritmo proposto

O algoritmo inicia sua execução criando um *layout* inicial através da distribuição aleatória das partículas no espaço bidimensional. Sendo assim, cada partícula possui uma posição inicial e, para possibilitar a evolução do enxame, uma velocidade inicial.

A cada iteração, e para cada partícula, a velocidade é atualizada de acordo com a Equação 26, que soma à velocidade anterior, a parte cognitiva da fórmula, representada pelo

conhecimento, e a parte social, representada pela colaboração entre as partículas. Nesta equação foi utilizado o fator de constrição que é, então, multiplicado ao restante da equação. A nova posição da partícula é determinada pela soma da sua posição atual e a nova velocidade, conforme a Equação 27.

$$\boldsymbol{v_i}^{\boldsymbol{it+1}} = k^{it} [{v_i}^{it} + {c_1}^{it} * \text{rand}()^{it} * ({pbest_i}^{it} - {x_i}^{it}) + {c_2}^{it} * \text{rand}()^{it} * ({gbest_i}^{it} - {x_i}^{it}) + {c_3}^{it} * \text{rand}()^{it} * (C^{it} - {x_i}^{it})] \quad (26)$$

$$\boldsymbol{x_i}^{\boldsymbol{it+1}} = {x_i}^{it} + \boldsymbol{v_i}^{\boldsymbol{it+1}} \quad (27)$$

onde:

k – fator de constrição

c_1, c_2, c_3 – parâmetros de confiança

rand() – função randômica [0, 1]

$pbest_i$ – melhor posição que a partícula i já obteve durante a busca

$gbest_i$ – posição da partícula i, quando obtida a melhor avaliação global do enxame

C – centróide do enxame

x_i – posição atual da partícula i

Nesta primeira proposta, antes da realização de todos os testes e comparações necessárias para verificação da viabilidade do método, foi testado também o componente inercial, aplicado à equação da velocidade, em substituição ao fator de constrição. Assim, a cada iteração, e para cada partícula, a velocidade é atualizada de acordo com a Equação 28, que multiplica o componente inercial pela velocidade anterior e, então, soma a parte cognitiva da fórmula, representada pelo conhecimento, e a parte social, representada pela colaboração entre as partículas.

$$\boldsymbol{v_i}^{it+1} = w^{it} * {v_i}^{it} + {c_1}^{it} * \text{rand}()^{it} * ({pbest_i}^{it} - {x_i}^{it}) + {c_2}^{it} * \text{rand}()^{it} * ({gbest_i}^{it} - {x_i}^{it}) + {c_3}^{it} * \text{rand}()^{it} * (C^{it} - {x_i}^{it}) \quad (28)$$

Comparando-se os resultados obtidos pelos dois parâmetros, verificou-se que o componente inercial gera melhores soluções para o problema de *layout*. Assim, este parâmetro foi utilizado ao longo deste trabalho.

Além de armazenar as velocidades $V_i = (v_1, v_2, ..., v_n)$, as posições $X_i = (x_1, x_2, ..., x_n)$ e as melhores posições já encontradas pelas partículas $Pbest_i = (pbest_1, pbest_2, ..., pbest_n)$, os

algoritmos armazenam também as posições já encontradas pelas partículas quando obteve-se o ótimo global $Gbest_i = (gbest_1, gbest_2, ..., gbest_n)$.

Sabe-se que, a cada iteração, após o cálculo da nova velocidade e novo posicionamento das partículas, aplica-se a função objetivo no enxame. Como o objetivo é minimizar, se o valor obtido for menor que o menor valor já encontrado pela função obtido significa que o movimento realizado pelas partículas ocasionou numa melhora do *layout*. Assim o *gbest* de cada partícula é ajustado de acordo com a posição em que as mesmas estavam quando se obteve o melhor resultado.

5.1.1 Aceleração por distância

O termo $(pbest_i - x_i)$, que compõem a equação para o cálculo da nova velocidade, se refere à distância entre a melhor posição já encontrada pela partícula *i* e a posição atual dessa mesma partícula. Tratando-se do problema de *layout*, o segundo termo teve que ser alterado e $(gbest_i - x_i)$ se refere à distância entre a posição da partícula quando o enxame atingiu o ótimo global e a posição atual da partícula *i*.

A melhor posição da partícula é calculada de acordo com a equação 29.

$$\text{Minimizar } F = \sum_{i=1}^{N} c_{ij} d_{ij} + P \sum_{i=1}^{N} f\left(\frac{d_{ij}}{t_{ij}}\right), \; j = 1, ..., N; \; j \neq i \qquad (29)$$

A primeira parte da Equação 29 faz o somatório de custo x distância da partícula *i* em relação às demais partículas do enxame e a segunda parte penaliza o resultado em caso de sobreposição das partículas. Caso o resultado seja menor que o resultado encontrado anteriormente, a posição da partícula é atualizada.

5.1.2 Limitador de velocidade

Além do componente inercial, o algoritmo proposto também controla a velocidade das partículas através do limitador v_{max}, para uma melhor exploração do espaço de busca.

Neste algoritmo o valor de v_{max} foi definido como 50. Assim, para v_{max} igual a 50, por exemplo, se o resultado da equação da velocidade for superior a 50 a velocidade da partícula assume valor igual a 50. Se o resultado for inferior a -50, a velocidade da partícula assume valor -50.

5.1.3 O algoritmo proposto: passo a passo

O algoritmo proposto neste trabalho segue os passos descritos no item 4.1.1.4, com algumas adaptações:

1. Cada partícula *i* representa uma facilidade. A posição inicial de cada partícula é escolhida aleatoriamente, para os eixos *x* e *y* do plano, assim como sua velocidade inicial. O *pbest* e *gbest* de cada partícula são inicializados com as posições iniciais das partículas;
2. Para cada partícula *i* é calculada a nova velocidade através da Equação 28 e a nova posição, pela Equação 27. O cálculo é realizado para os eixos *x* e *y*;
3. A função objetivo de cada partícula *i* é calculada através da equação 29.
4. O valor da avaliação da partícula *i*, através da Equação 29, é comparado ao melhor (menor) valor já obtido por esta partícula. Se o valor for melhor, o $pbest_i$ é atualizado. O *pbest* da partícula *i*, representa a melhor posição que a partícula *i* já ocupou no espaço em relação ao enxame;
5. O valor da avaliação do enxame, calculado com a Equação 8, é comparado ao melhor (menor) valor já obtido pelo enxame. Se o valor for melhor, o $gbest_i$ é atualizado. O *gbest* da partícula *i* é a posição que essa partícula ocupou no espaço, quando o melhor *layout* foi obtido;
6. Repetir os passos anteriores a partir do item 2, para todas as partículas, e o algoritmo até que o critério de parada seja satisfeito, neste trabalho definido pelo número de iterações.

Considerando-se que cada partícula armazena o seu melhor local (*pbest*) e seu melhor global (*gbest*) para migrar para uma posição mais promissora, a Figura 5 apresentada por Prado e Saramago (2005) é modificada e a Figura 5 traduz a proposta deste novo algoritmo.

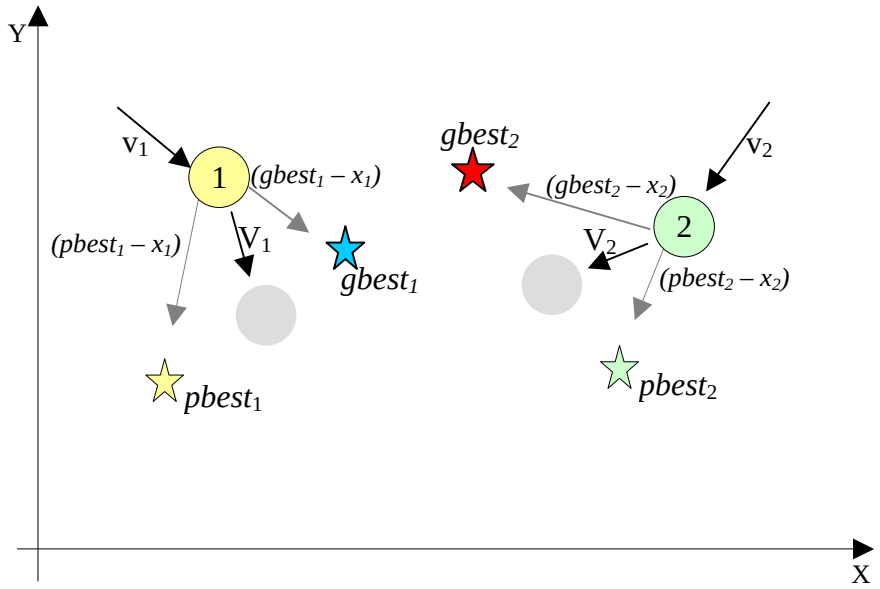


Figura 5 – Comportamento das partículas no espaço de busca bidimensional conforme o algoritmo proposto na abordagem I

onde:

$pbest_i$ – melhor posição da partícula *i*

$gbest_i$ – melhor posição da partícula *i* quando se obteve o melhor *layout*

$(pbest_i – x_i)$ – distância entre a melhor posição da partícula *i* e sua posição atual.

$(gbest_i – x_i)$ – distância entre a melhor posição da partícula *i* quando se obteve o melhor *layout* e sua posição atual

◯ – posição atual

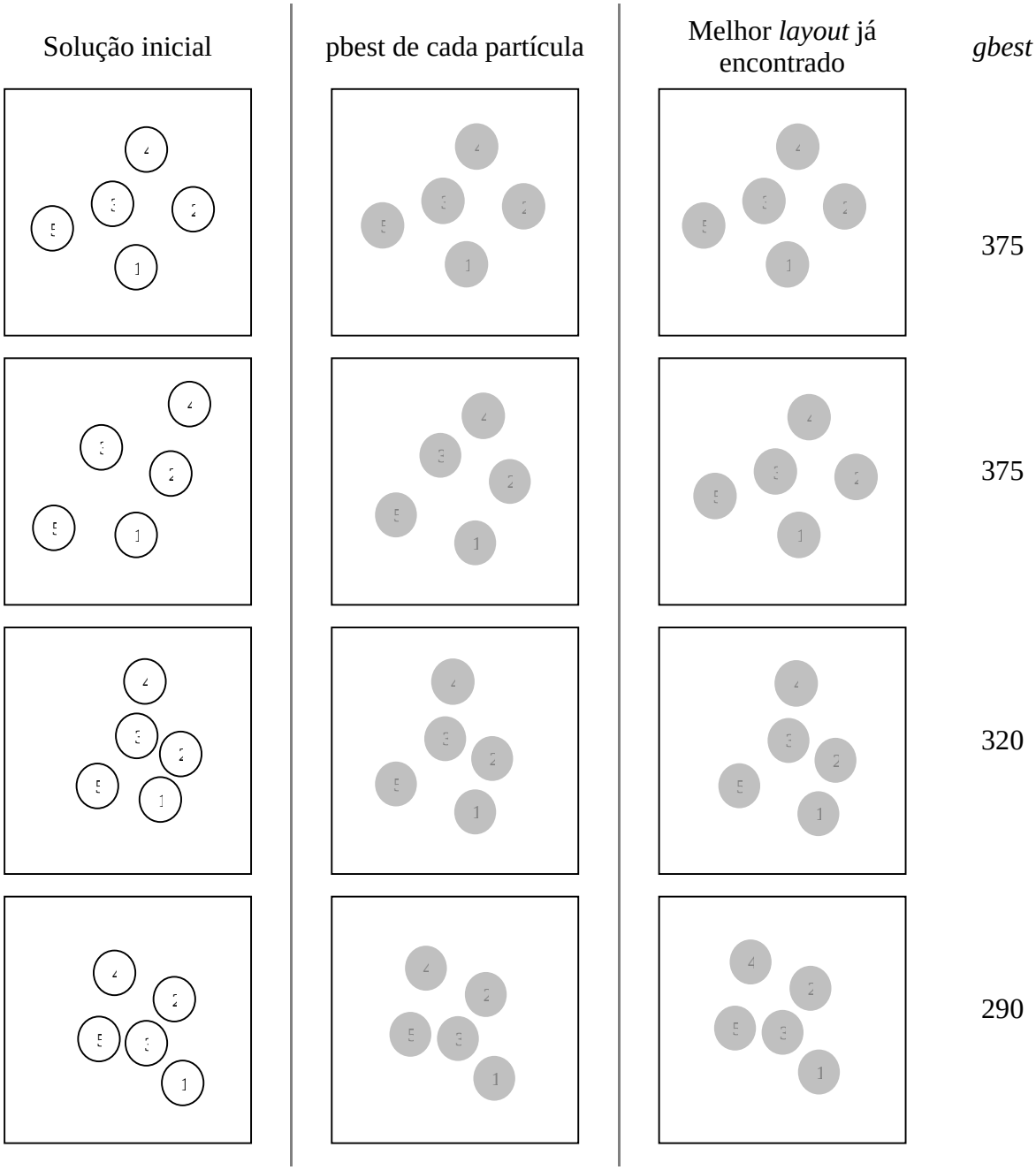
● – próxima posição

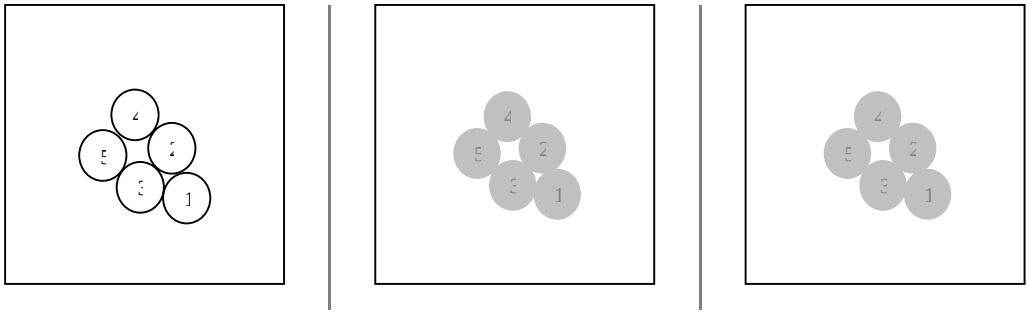
A posição na qual a partícula se encontra foi calculada mediante as equações 28 e 27 e, para o cálculo da nova posição, indicada pelo círculo sombreado, as mesmas equações são utilizadas, levando-se em consideração a velocidade anterior da partícula, *v*, a sua posição atual (x, y), a melhor posição já visitada pela partícula, *pbest* e a melhor posição já visitada pela partícula, em relação ao enxame, *gbest*, como mostra a Figura 5.

A seguir é apresentada um exemplo ilustrativo de 5 partículas no espaço de busca cujos custos estão relatados na Tabela 7. Neste caso, foram necessárias 7 iterações para obtenção de um *layout* de qualidade. A cada iteração o *pbest* das partículas e o melhor *layout* já encontrado são mostrados.

Tabela 7 – Exemplo de uma tabela de custos entre 5 facilidades

	1	**2**	**3**	**4**	**5**
1	0	5	0	3	0
2	5	0	1	10	3
3	0	1	0	1	5
4	3	10	1	0	10
5	0	3	5	10	0





Percebe-se na execução que, nem sempre o movimento das partículas conduz as mesmas a ocuparem melhores posições ou mesmo numa melhora do *layout* final.

5.2 Resultados

Para avaliar a eficiência do método enxame de partículas, aplicada ao problema de *layout*, utilizando o método AR, os resultados foram comparados com os resultados obtidos por Anjos e Vannelli (2002). Foi realizado também um comparativo em relação à utilização do fator de constrição (*k*), conforme mostra a Tabela 9 e do componente inercial (*w*), conforme Tabela 10.

Para esta proposta, os parâmetros para os fatores de confiança foram inicializados com os seguintes valores: c_{1ini} = 7,5, c_{1fin} = 2,0, c_{2ini} = 7,5, c_{2fin} = 2,0, c_{3ini} = 2,0 e c_{3fin} = 0,01, com C > 4. O algoritmo utilizou os problemas teste de Nugent *et al* (1968) e, para cada problema foram realizadas 10 execuções, sendo o critério de parada estabelecido em 1000 iterações.

Para cada *layout* gerado, o percentual de sobreposição foi calculado de acordo com a Equação 30 (CASTILLO E SIM, 2003):

$$\%\ \text{sobreposição} = \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\max\left\{0, r_i + r_j - d_{ij}\right\}}{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(r_i + r_j)} \qquad (30)$$

Tabela 8 – Resultados obtidos para facilidades com mesma área, utilizando-se o fator de constrição

Número de facilidades	Enxame de Partículas				Anjos e Vannelli (2002)		
	Resultado	**Sobreposição (%)**	**α**	**Tempo (s)**	**Resultado**	**Sobreposição (%)**	**α**
12	786	0,0	4	0,51	528,9	0,0	0,005
15	1586	0,0	2	0,78	1154,0	0,0	0,005
20	2964	0,0	2	0,92	3064,8	0,0	0,005
30	8352	0,0	1	1,15	9208,0	0,0	0,005

Tabela 9 – Resultados obtidos para facilidades com mesma área, utilizando-se o componente inercial

Número de facilidades	Enxame de Partículas				Anjos e Vannelli (2002)		
	Resultado	**Sobreposição (%)**	**α**	**Tempo (s)**	**Resultado**	**Sobreposição (%)**	**α**
12	636	0,0	2	0,76	528,9	0,0	0,005
15	1210	0,0	1	0,75	1154,0	0,0	0,005
20	2710	0,0	0,005	0,87	3064,8	0,0	0,005
30	8274	0,0	0,005	1,34	9208,0	0,0	0,005

Comparando-se os resultados obtidos nas tabelas 8 e 9 observa-se que a utilização do parâmetro *w* gera melhores *layouts* e, por esta razão, este parâmetro foi utilizado no restante deste trabalho. Observa-se também que o método enxame de partículas encontra melhores resultados para grandes instâncias do problema, ou seja, acima de 20 facilidades. A Figura 6 mostra o *layout* gerado para 30 facilidades.

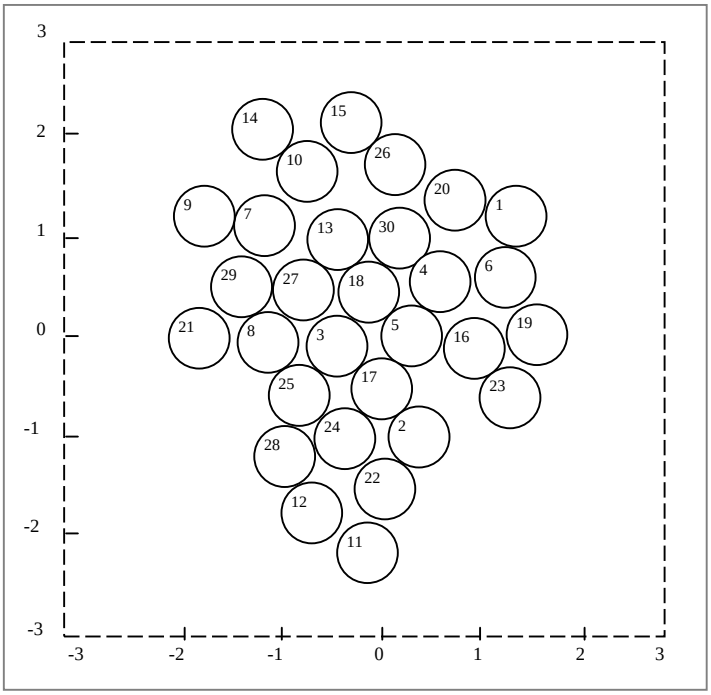


Figura 6 – *Layout* gerado para 30 facilidades de mesma área

Não foi possível comparar os tempos de execução, pois Anjos e Vannelli (2002) apenas relataram que as soluções obtidas por eles foram encontradas em menos de 2 segundos. Comprovando ainda mais a eficiência do método enxame de partículas, os resultados foram obtidos em menos de 1 segundo, exceto para o problema envolvendo 30 facilidades. Vale salientar também que, para todas as instâncias testadas, 500 iterações são suficientes para se obter bons resultados. Em alguns casos, o melhor *layout* foi encontrado antes mesmo da 100ª iteração.

Após vários testes, observou-se que o parâmetro α influencia significativamente na obtenção de um *layout* de qualidade. Pode-se observar na Tabela 10 que, para 12 facilidades, a média dos resultados, tempos e iteração de 20 execuções do algoritmo, são semelhantes. Porém, à medida que o valor de α diminui, começa a ocorrer sobreposição de facilidades em alguns *layouts* e a porcentagem dessa sobreposição aumenta consideravelmente.

Tabela 10 – Ocorrência de sobreposição das facilidades de acordo com o valor de α - resultados obtidos para 12 facilidades de mesma área

α	**Resultado médio**	**Tempo médio (s)**	**Iteração média**	**Número de *layouts* com sobreposição**	**Média de sobreposição (%)**
4	853	0,72	161	0	0,0
2	862	0,49	198	0	0,0
1	868	0,71	132	5	0,2
0,01	863	0,59	117	11	1,1
0,005	876	0,67	177	12	2,0

O mesmo estudo foi realizado para 30 facilidades. A Tabela 11 mostra as médias dos resultados obtidos em 10 execuções do algoritmo. Utilizando-se $\alpha \leq 1$ a maioria dos *layouts* gerados apresenta sobreposição e, quanto menor o valor de α, maior a porcentagem de sobreposição. Com $\alpha = 2$, o número de *layouts* com sobreposição é menor, algumas vezes nulo.

Tabela 11 – Ocorrência de sobreposição das facilidades de acordo com o valor de α - resultados obtidos para 30 facilidades de mesma área

α	**Resultado médio**	**Tempo médio (s)**	**Iteração média**	**Número de *layouts* com sobreposição**	**Média de sobreposição (%)**
4	11185	1,47	66	0	0,0
2	11061	1,47	86	3	0,3
1	10828	1,47	78	8	1,5
0,01	10190	1,46	62	10	0,5
0,005	9700	1.46	64	9	0,3

Ainda em relação à Tabela 11, com $\alpha = 4$ não há sobreposição em 100% dos *layouts*, no entanto, observa-se que algumas facilidades ficam distantes das demais. O interessante é que isso não ocorre para 12 e 15 facilidades, que mesmo utilizando $\alpha = 4$, são obtidos *layouts* de qualidade, como mostra a Figura 7. Em todos os testes foi utilizado $P = 10$.

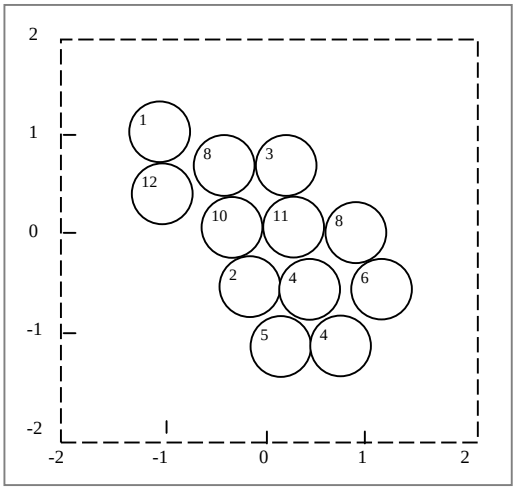


Figura 7 – *Layout* gerado para 12 facilidades de mesma área – Abordagem I

Comparando-se as tabelas 11 e 12 observa-se que, para 30 facilidades, os melhores *layouts* são obtidos com valor de α menor, enquanto que, para 12 facilidades, ocorre o inverso, ou seja, quanto maior o valor de α, melhores *layouts* são gerados. Outro resultado importante é que, para 30 facilidades, o melhor *layout* é encontrado antes daqueles encontrados para 12 facilidades.

6 IMPLEMENTAÇÃO DO MÉTODO ENXAME DE PARTÍCULAS PARA O PROBLEMA DE *LAYOUT* E RESULTADOS – ABORDAGEM II

A segunda abordagem utiliza vários *layouts* simultaneamente para a obtenção do *layout* final. Para a implementação deste algoritmo foram necessárias algumas alterações, principalmente em relação ao *gbest* que consiste no melhor dentre todos os *layouts* envolvidos no processo, ou seja, aquele com menor *pbest*. Cada partícula *i* do *gbest* assume o valor da sua posição – $gbest_i$.

6.1 O algoritmo

Após determinar a quantidade de *layouts*, $s = 1, 2, ..., m$, e de partículas a serem utilizadas e o número de iterações necessárias para execução, o algoritmo é inicializado. Em cada *layout*, as partículas são posicionadas no espaço bidimensional (x, y), de forma aleatória e recebem um valor inicial para sua velocidade, o que constitui a solução inicial.

A cada iteração *it*, e para cada *layout*, a velocidade é atualizada de acordo com a Equação 31, que soma à velocidade anterior, a parte cognitiva da fórmula, representada pelo conhecimento, e a parte social, representada pela colaboração entre as partículas. A nova posição da partícula é determinada pela soma da sua posição atual e a nova velocidade, em cada *layout*, conforme a Equação 32.

$$\boldsymbol{v_{is}}^{it+1} = v_{is}{}^{it} + c_1{}^{it} * \text{rand}()^{it} * (pbest_{is}{}^{it} - x_{is}{}^{it}) + c_2{}^{it} * \text{rand}()^{it} * (gbest_i{}^{it} - x_{is}{}^{it}) \quad (31)$$

$$x_{is}{}^{it+1} = x_{is}{}^{it} + \boldsymbol{v}_{is}{}^{it+1} \quad (32)$$

A cada iteração, e para cada *layout*, a velocidade é atualizada de acordo com a Equação 33, que multiplica o componente inercial e a velocidade anterior e soma à parte cognitiva da fórmula, representada pelo conhecimento, e à parte social, representada pela colaboração entre as partículas.

$$\boldsymbol{v}_{is}^{\,it+1} = w^{\,it} * v_{is}^{\,it} + c_1^{\,it} * \text{rand}()^{it} * (pbest_{is}^{\,it} - x_{is}^{\,it}) + c_2^{\,it} * \text{rand}()^{it} * (gbest_i^{\,it} - x_{is}^{\,it}) \qquad (33)$$

O algoritmo armazena as velocidades das partículas $V_i = (v_1, v_2, ..., v_n)$, as posições das partículas $X_i = (x_1, x_2, ..., x_n)$, as posições das partículas quando encontrado o melhor *pbest* dentre todos os *layouts* $Pbest_{is} = (pbest_{1s}, pbest_{2s}, ..., pbest_{nm})$ e as posições das partículas do *layout* que apresentar o *pbest* menor daquele já armazenado $Gbest_i = (gbest_1, gbest_2, ..., gbest_n)$.

O *pbest* é o resultado da aplicação da função objetivo a cada *layout*. O menor *pbest* é considerado *gbest*, ou seja, o melhor *layout* já encontrado. Sempre que *pbest* encontra um valor menor que *gbest*, este é atualizado. A figura 8 mostra um exemplo ilustrativo de 4 partículas e 4 *layouts*.

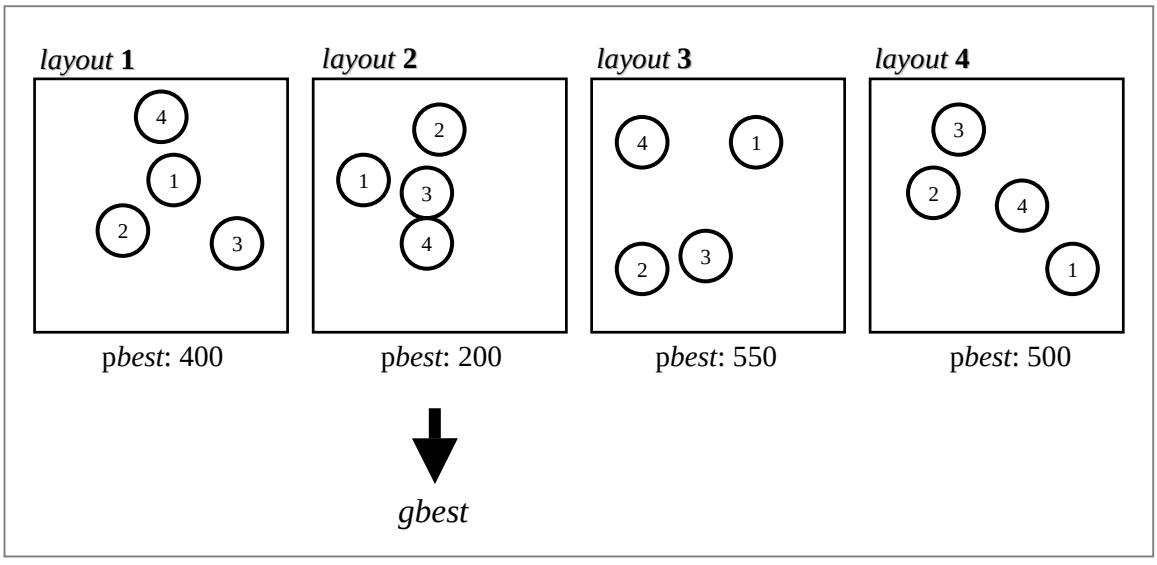


Figura 8 – Determinação do *gbest* no exemplo ilustrativo de 4 partículas e 4 *layouts*

6.1.1 Parâmetros de confiança

Para esta proposta foram usadas as Equações 16 e 17 propostas por Ratnaweera *et al* (2004) para variar os parâmetros de confiança, c_1 e c_2.

6.1.2 Aceleração por distância

O termo ($pbest_i - x_i$), que compõem a equação para o cálculo da nova velocidade, se refere à distância entre a melhor posição já encontrada pela partícula *i* e a posição atual dessa mesma partícula, e ($gbest_i - x_i$), à distância entre a posição da partícula *i* quando se obteve o melhor *layout* e a posição atual da partícula.

6.1.3 Limitador de velocidade

Além do componente inercial (*w*), o algoritmo proposto controla a velocidade das partículas através do limitador v_{max}, para uma melhor exploração do espaço de busca.

Neste algoritmo o valor de v_{max} variou de acordo com o número de partículas testadas, variando de 50 a 250. Para v_{max} igual a 100, por exemplo, se o resultado da equação da velocidade for superior a 100 a velocidade da partícula assume valor igual a 100. Se o resultado for inferior a -100, a velocidade da partícula assume valor -100.

6.1.4 O algoritmo proposto: passo a passo

Inicialmente, s = 1, 2, ..., *m layouts* são gerados, onde cada *layout* é formado por *N* facilidades. Cada facilidade é representada por uma partícula *i*, no enxame de partículas.

1. A posição inicial de cada partícula, p_i, é escolhida aleatoriamente, assim como sua velocidade inicial, v_i. Para cada partícula *i*, em cada um dos *layouts*, é calculada a nova velocidade através da Equação 33 e a nova posição, pela equação 32.

3. A função objetivo, F_s, de cada *layout* é calculada através da Equação 8.

4. Para cada *layout s*, se $F_s < pbest_s$, então atualize $pbest_s$ com o valor de F_s.

5. Calcule $gbest^{it}$. $gbest^{it} = min\ \{pbest_1^{it}, pbest_2^{it}, ..., pbest_n^{it}\}$. Se $gbest^{it} < gbest$, então atualize *gbest* com o valor de $gbest^{it}$ e atualize cada $gbest_i$ com a posição de cada partícula *i*.

6. Repetir a partir do item 2, até que $it = it_{max}$, sendo it_{max} o número máximo de iterações

definidas.

6.2 Representação do *layout*

Assim como Tam (2002), este trabalho fez uso da estrutura de árvore binária para representar os *layouts* retangulares gerados, cujas facilidades estão representadas pelas folhas da árvore. Considerando-se o exemplo de 12 facilidades, inicialmente é calculado o *gbest* e uma floresta com 12 árvores é gerada. A cada iteração duas árvores são unidas até a formação da árvore final e para uni-las é considerada a menor razão da distância euclidiana pela soma de seus diâmetros.

Para gerar o *layout* retangular é realizada uma busca na árvore binária, da esquerda para a direita. O primeiro nó esquerdo da subárvore esquerda é analisado e as áreas de todas as suas folhas são somadas para gerar o primeiro particionamento do *layout*. Da mesma forma se existir um nó à esquerda do anterior. Quando não houver mais nós à esquerda, o algoritmo verifica os nós à direita. Pode acontecer, porém, de um nó da direita possuir nó à esquerda, sendo estes analisados primeiramente. Após percorrer todos os nós e folhas da subárvore esquerda, a busca é realizada na subárvore direita à raiz.

6.3 Resultados

Foram realizados testes com instâncias cujas facilidades possuem áreas de tamanho igual e testes com facilidades de áreas diferentes.

6.3.1 Facilidades de mesma área

Para facilidades de mesma área o valor de α variou de acordo com o número de facilidades testadas e com os testes obtidos e apresentados pelas tabelas 9 e 10. Os valores dos parâmetros utilizados também variaram de acordo com o número de facilidades. Em facilidades de mesma área foi utilizado $r = 1$ para todas as facilidades.

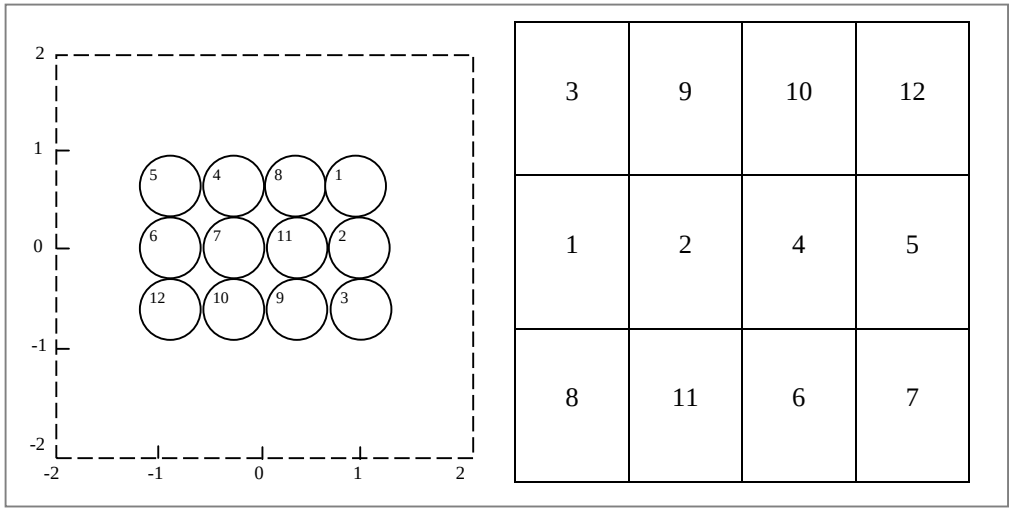


Figura 9 – *Layout* gerado para 12 facilidades de mesma área – Abordagem II

Para a primeira instância testada, 12 facilidades, conforme Tabela 12, utilizou-se $\alpha = 2$ e, para os parâmetros c_1 e c_2, 2,5 como valor inicial e 0,9 como valor final. O componente inercial variou de 0,9 a 0,6 e v_{max} foi definido como 150. Para este teste foram utilizadas 1000 *layouts* e 1000 iterações. Para a penalização no caso de sobreposição foi definido $P = 10$. A Figura 9 mostra o *layout* gerado com essa configuração.

Tabela 12 – Resultados obtidos para 12, 15, 20 e 30 facilidades de mesma área, utilizando-se vários *layouts*

Número de facilidades	**Enxame de Partículas**				**Anjos e Vannelli (2002)**		
	Resultado	**Sobreposição (%)**	**α**	**Tempo (s)**	**Resultado**	**Sobreposição (%)**	**α**
12	542.2	0,0	2	38.5	528,9	0,0	0,005
15	1057.1	0,0	2	54.2	1154,0	0,0	0,005
20	2407.4	0,0	1	86.3	3064,8	0,0	0,005
30	5871.9	0,0	1	176.6	9208,0	0,0	0,005

Em relação ao *layout* gerado para 12 facilidades de mesma área, a Tabela 13 mostra a relação de custos que deve haver entre as facilidades (NUGENT *et al*, 1968) e a Tabela 14 mostra as distâncias entre as facilidades. Quanto maior o custo entre as facilidades menor deve ser a distância entre elas.

Tabela 13 – Relação de custos entre 12 facilidades

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	5	2	4	1	0	0	6	2	1	1	1
2		0	3	0	2	2	2	0	4	5	0	0
3			0	0	0	0	0	5	5	2	2	2
4				0	5	2	2	10	0	0	5	5
5					0	10	0	0	0	5	1	1
6						0	5	1	1	5	4	0
7							0	10	5	2	3	3
8								0	0	0	5	0
9									0	0	10	5
10										0	5	0
11											0	2
12												0

Tabela 14 – Distância entre as facilidades do *layout* gerado para 12 facilidades de mesma área

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	2.0	4.0	4.0	6.0	6.3	4.4	2.0	4.4	5.6	2.8	7.2
2		0	2.0	4.4	6.3	6.0	4.0	2.8	2.8	4.4	2.0	6.3
3			0	5.6	7.2	6.3	4.4	4.4	2.0	4.0	2.8	6.0
4				0	2.0	2.8	2.0	2.0	4.4	4.0	2.8	4.4
5					0	2.0	2.8	4.0	5.6	4.4	4.4	4.0
6						0	2.0	4.4	4.4	2.8	4.0	2.0
7							0	2.8	2.8	2.0	2.0	2.8
8								0	4.0	4.4	2.0	5.6
9									0	2.0	2.0	4.0
10										0	2.8	2.0
11											0	4.4
12												0

Verifica-se que a maioria das facilidades satisfaz a relação custo *versus* distância. Por exemplo, todas as facilidades com custo igual a 10, ressaltado na Tabela 13, estão próximas umas das outras, exceto as facilidades 7 e 8. Vale salientar que 2 é a distância mínima que pode haver entre as facilidades pois o raio é 1.

Para 15 facilidades, o valor inicial para c_1 e c_2 ficou definido como 1,5 e 0,9 como valor final dessas variáveis. O componente inercial variou de 0,9 a 0,6 e v_{max} foi definido como 100. Para este teste foram utilizadas 1000 partículas e 1000 iterações. Para a

penalização no caso de sobreposição foi definido P = 20. A Figura 10 mostra um *layout* gerado com essa configuração.

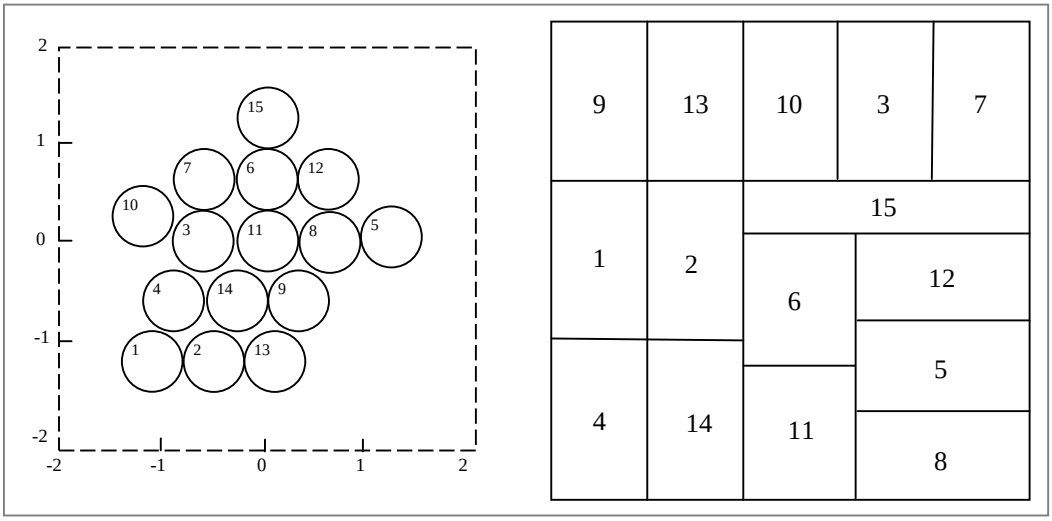


Figura 10 – *Layout* gerado para 15 facilidades de mesma área – Abordagem II

Para 20 facilidades, o valor inicial para c_1 e c_2 ficou definido como 1,5 e 0,9 como valor final dessas variáveis. O componente inercial variou de 0,9 a 0,4 e v_{max} foi definido como 250. Para este teste foram utilizadas 1000 partículas e 1000 iterações. Para a penalização no caso de sobreposição foi definido P = 200. A Figura 11 mostra o *layout* gerado com essa configuração.

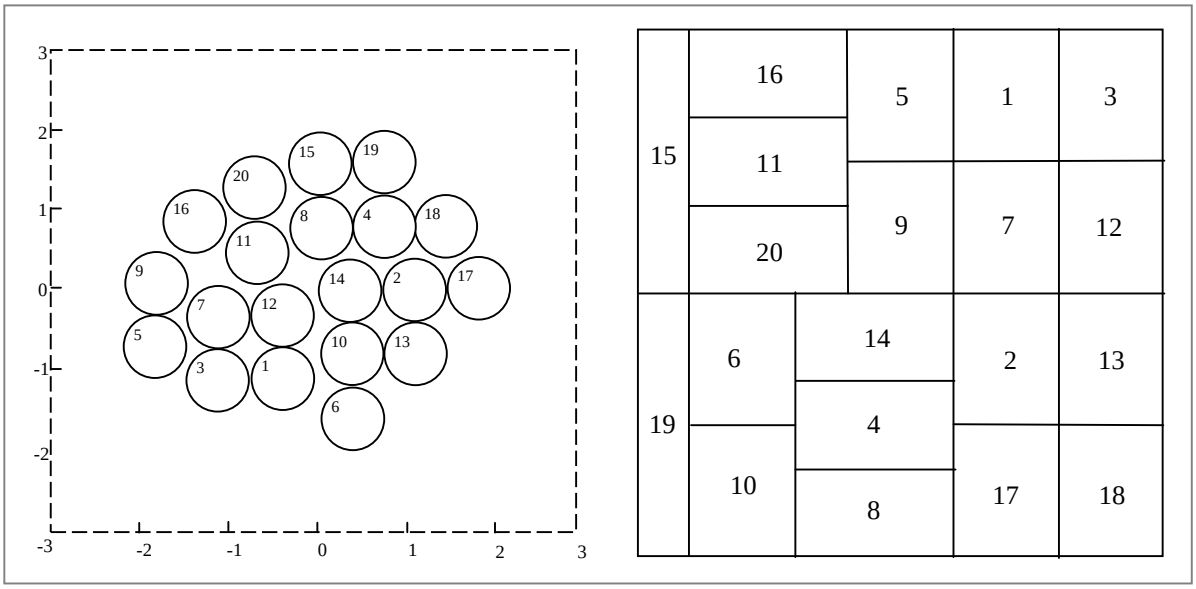


Figura 11 – *Layout* gerado para 20 facilidades de mesma área – Abordagem II

Para 30 facilidades, o valor inicial para c_1 e c_2 ficou definido como 1,5 e 0,9 como valor final dessas variáveis. O componente inercial variou de 0,9 a 0,4 e v_{max} foi definido como 250. Para este teste foram utilizadas 1000 partículas e 1000 iterações. Para a penalização no caso de sobreposição foi definido P = 500. A Figura 12 mostra o *layout* gerado com essa configuração.

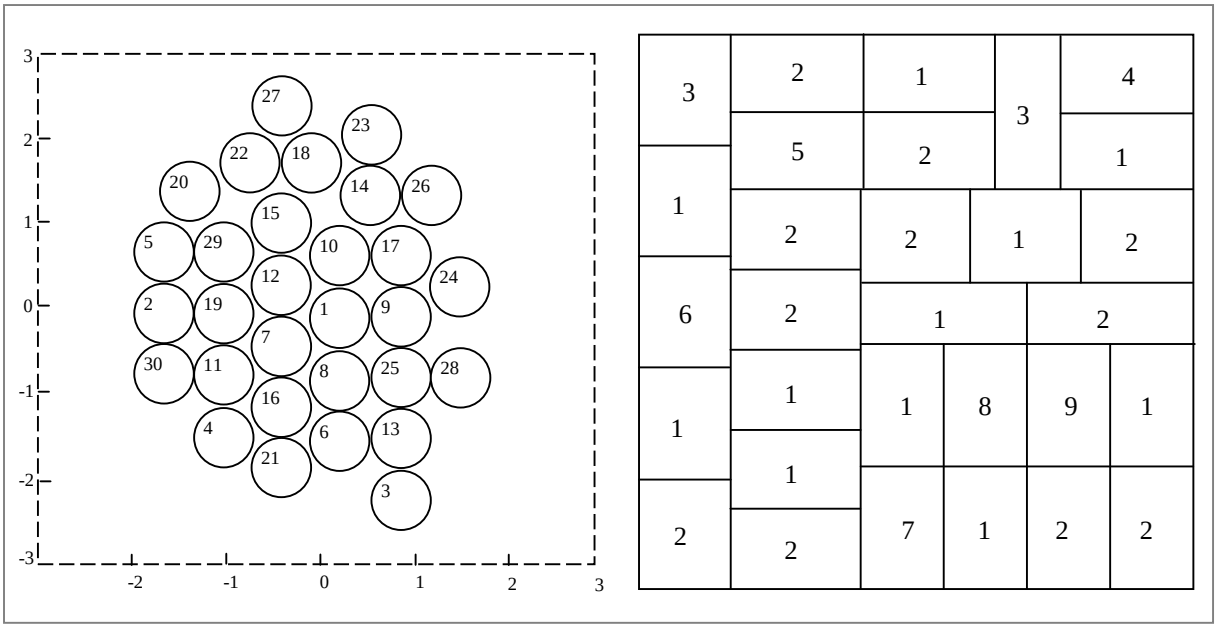


Figura 12 – *Layout* gerado para 30 facilidades de mesma área – Abordagem II

A tabela 15 mostra os parâmetros utilizados para cada uma das instâncias analisadas. Observa-se que o parâmetro *P*, utilizado junto à equação da função objetivo quando ocorre sobreposição entre partículas, varia em relação ao número de facilidades e é precisa ser maior quanto maior o enxame.

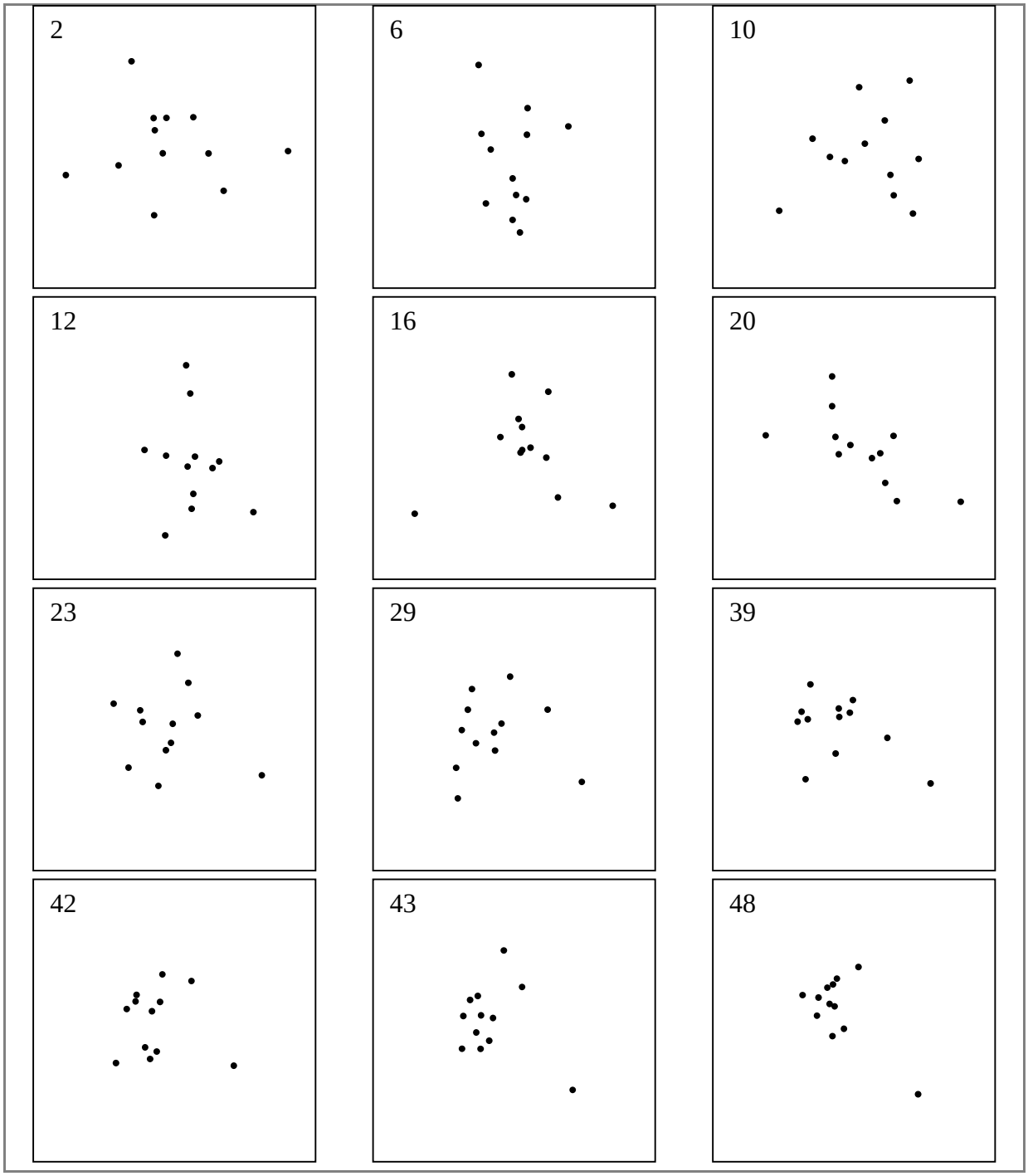
Tabela 15 – Variação do fator de penalidade (*P*)

Parâmetros Facilidades	c_{1ini}	c_{1fin}	c_{2ini}	c_{2fin}	w_{ini}	w_{fin}	v_{max}	P
12	2,5	0,9	2,5	0,9	0,9	0,6	150	10
15	1,5	0,9	1,5	0,9	0,9	0,6	100	20
20	1,5	0,9	1,5	0,9	0,9	0,4	250	200
30	1,5	0,9	1,5	0,9	0,9	0,4	250	500

Analisando a tabela acima, verifica-se também que o valor que limita a velocidade das partículas aumenta em função do número de facilidades. Isso ocorre, pois quanto maior o

número de partículas, maior a necessidade de exploração do espaço e troca de informações entre elas a fim de gerar *layouts* de qualidade.

A Figura 13 mostra parte de uma execução do algoritmo. Cada quadro representa o posicionamento das partículas quando houve melhora no resultado da função objetivo, ou seja, quando o *gbest* das partículas foi atualizado. A numeração indica a iteração na qual isso aconteceu.



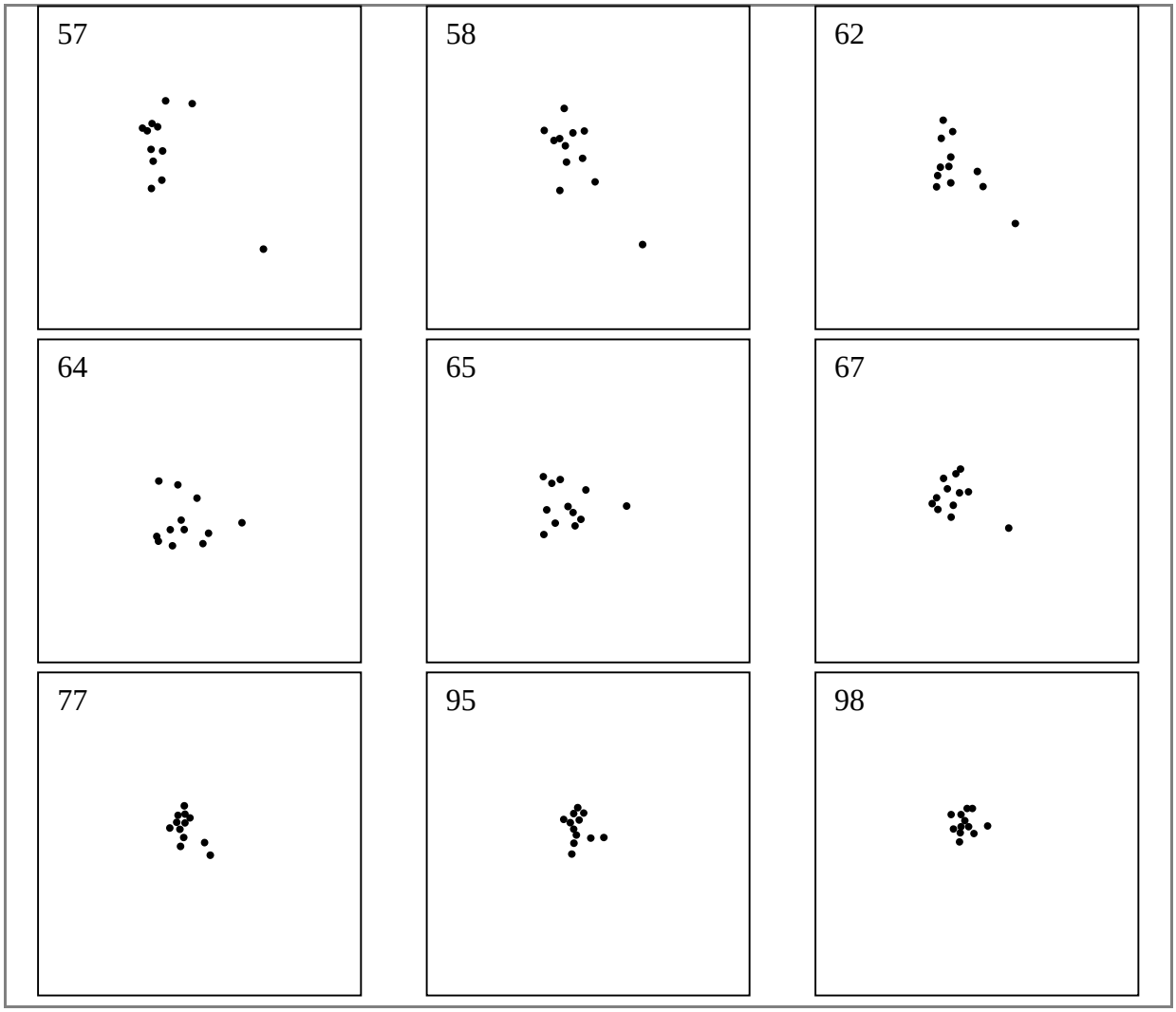


Figura 13 – Representação parcial da movimentação das partículas no espaço de busca para formação de um *layout*

6.3.2 Facilidades de áreas diferentes

Para a obtenção de *layouts* de qualidade utilizando facilidades com áreas diferentes foi necessária a realização de vários testes no que diz respeito à escolha dos valores a serem atribuídos às variáveis. As áreas das facilidades estão descritas na tabela 16.

Tabela 16 – Área das facilidades, para o cálculo do raio

Facilidade	Área	Facilidade	Área	Facilidade	Área
1	8	11	10	21	3
2	10	12	2	22	2
3	5	13	8	23	1
4	6	14	6	24	9
5	6	15	5	25	8
6	2	16	1	26	5
7	1	17	9	27	4
8	8	18	1	28	1
9	4	19	4	29	6
10	8	20	6	30	1

Para o cálculo do raio utilizou-se a Equação 34 (CASTILLO E SIM, 2003):

$$r_i = \frac{\sqrt{a}_i}{2}, \quad \text{i} = 1, .., N \tag{34}$$

Tabela 17 – Resultados obtidos para 12, 15, 20 e 30 facilidades de áreas diferentes

Número de facilidades	Enxame de Partículas				Anjos e Vannelli (2002)		
	Resultado	Sobreposição (%)	α	Tempo (s)	Resultado	Sobreposição (%)	α
12	626.1	0.0	2	37.7	632.6	0.6	0,005
15	1241.3	0.0	2	53.9	1389.9	0.2	0,005
20	2704.8	0.0	1,5	88.6	3605.2	0.0	0,005
30	6047.2	0.0	1,5	183.5	10516.3	0.0	0,005

A Tabela 17 mostra os resultados obtidos pelo método enxame de partículas. Para todas as instâncias testadas, os resultados foram melhores se comparados aos obtidos por Anjos e Vannelli (2002), principalmente pelo fato de nenhum dos *layouts* gerados apresentarem sobreposição. A abordagem utilizada implicou em maior tempo de processamento.

A Figura 14 mostra o *layout* gerado para 12 facilidades de áreas diferentes. O valor inicial para c_1 e c_2 foi definido como 1,5 e 0,5 como valor final. O componente inercial variou de 0,9 a 0,4 e v_{max} foi definido como 150. Neste teste foram utilizadas 1000 partículas e 1000 iterações. Para penalizar a função objetivo no caso de sobreposição foi usado $P = 10$.

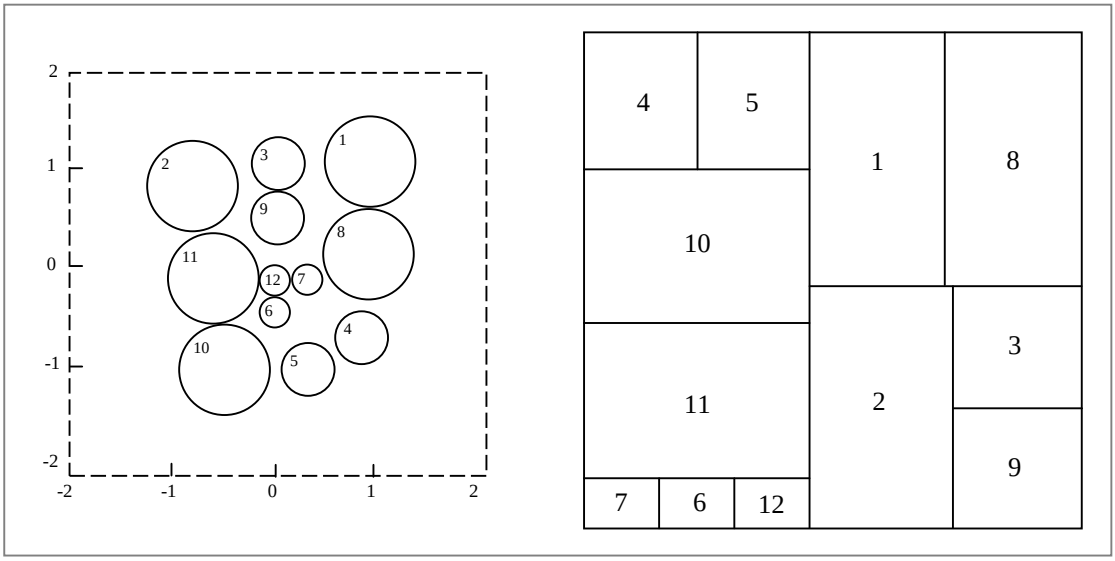


Figura 14 – *Layout* gerado para 12 facilidades de áreas diferentes – Abordagem II

Após vários testes, observou-se que, para 12 facilidades, a atribuição de valores muito altos para os parâmetros c_1, c_2 e w_{ini} pode ocasionar sobreposição das facilidades assim como valores muito baixos para o limitador de velocidade v_{max}. E ainda, a diferença entre w_{ini} e w_{fin} deve ser aproximadamente 0,5.

Mesmo que o tempo de processamento seja maior é importante utilizar um número maior de *layouts* para execução do algoritmo. Quanto ao número de iterações, não há necessidade deste ser superior a 2000.

Tabela 18 – Distância entre as facilidades do *layout* gerado para 12 facilidades de áreas diferentes

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	6.0	3.0	6.0	7.2	5.8	4.4	3.0	3.6	8.6	6.4	5.0
2		0	3.1	7.8	7.2	5.0	5.0	6.3	3.1	6.0	3.1	4.2
3			0	6.7	7.0	5.0	4.1	4.2	2.0	7.2	4.4	4.0
4				0	2.2	3.1	2.8	3.0	5.0	5.0	5.3	3.6
5					0	2.2	3.0	4.4	5.0	3.0	4.2	3.1
6						0	1.4	3.6	3.0	2.8	2.2	1.0
7							0	2.2	2.2	4.2	3.0	1.0
8								0	3.1	6.4	5.0	3.1
9									0	5.3	2.8	2.0
10										0	3.0	3.6
11											0	2.0
12												0

Verifica-se que a maioria das facilidades satisfaz a relação custo *versus* distância. Por exemplo, todas as facilidades com custo igual a 10, ressaltado na Tabela 13, estão próximas umas das outras.

A distância mínima entre duas facilidades é determinada pela soma de seus raios ou, conforme a Tabela 19, caso duas facilidades tenham o mesmo diâmetro a distância mínima que pode ter entre elas é o valor do diâmetro.

Tabela 19 – Diâmetro para 12 facilidades

Facilidade	1	2	3	4	5	6	7	8	9	10	11	12
Diâmetro	3	3	2	2	2	1	1	3	2	3	3	1

De acordo com a Tabela 19 é possível verificar a proximidade ideal entre as facilidades. De acordo com o *layout* gerado para 12 facilidades de áreas diferentes (Figura 14) e as distâncias entre elas (Tabela 18) é possível verificar a viabilidade do *layout*.

A Figura 15 mostra o *layout* gerado para 20 facilidades de áreas diferentes, sendo estas de tamanhos diferentes. O valor inicial para c_1 e c_2 foi definido como 1,5 e 0,5 como valor final. O componente inercial variou de 0,9 a 0,4 e v_{max} foi definido como 250. Neste teste foram utilizadas 1000 partículas e 1000 iterações. Para penalizar a função objetivo no caso de sobreposição foi usado $P = 100$.

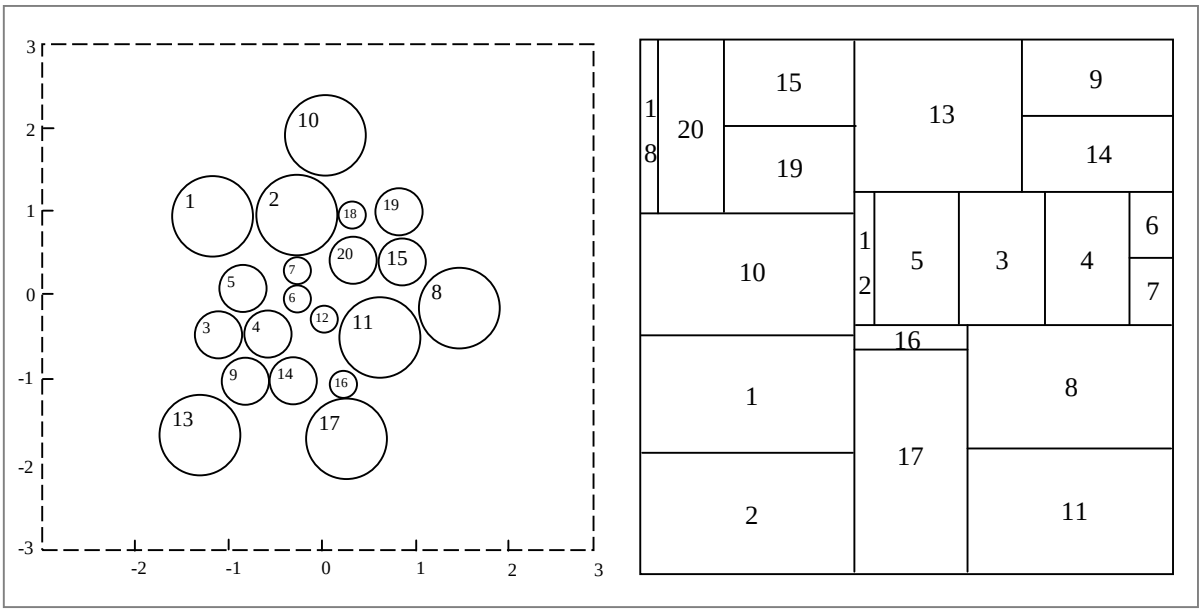


Figura 15 – *Layout* gerado para 20 facilidades de áreas diferentes – Abordagem II

CONCLUSÃO

Este trabalho propôs o uso do método Enxame de Partículas na otimização do problema de *layout*. Para a modelagem do problema foi utilizada o modelo *Attractor-Repeller* (AR) para evitar a sobreposição das facilidades, restrição considerada neste trabalho. Junto à equação da velocidade do método foram testados dois parâmetros: o componente inercial e o fator de restrição e, após vários testes, constatou-se a eficiência do primeiro para o problema de *layout* e este foi usado nos dois algoritmos desenvolvidos neste trabalho.

O primeiro algoritmo desenvolvido está baseado na idéia original do método enxame de partículas, porém com algumas modificações para se adequar ao problema de *layout*. A alteração feita no *gbest* é uma delas. Neste algoritmo, além de cada partícula armazenar o *pbest*, cada partícula armazena também seu próprio *gbest*, o que permite que as partículas encontrem as melhores posições no enxame, definindo, assim, o *layout*. Outra modificação realizada foi a introdução do centróide na equação da velocidade, o que auxiliou para evitar a dispersão das partículas no espaço de busca.

Foram realizados vários testes do algoritmo e verificou-se a importância do parâmetro α, que compõem a equação da distância alvo do modelo AR e realiza uma força repulsiva quando há sobreposição das partículas. Por esta razão foram testados vários valores para este parâmetro, para todas as instâncias do problema, na busca por bons resultados. Verificou-se que quanto menor o valor de α, maior a incidência de sobreposição das facilidades no *layout* e, que o valor escolhido para este parâmetro deve ser maior quanto maior o número de facilidades.

Os resultados obtidos por este algoritmo foram satisfatórios para mais de 20 facilidades, se comparados com os autores do modelo, Anjos e Vannelli (2002). Isso é surpreendente, pois a maior dificuldade, por tratar-se um problema NP, é encontrar bons resultados para grandes instâncias do problema. O método enxame de partículas provou sua eficácia.

O segundo algoritmo desenvolvido neste trabalho utiliza vários *layouts* para se obter um *layout* de qualidade. Neste caso, cada *layout* tem um *pbest* e o melhor deles, ou seja, o *layout* com o menor valor de *pbest* é considerado o *gbest* e as partículas desse *layout* assumem suas posições como *gbest* que são utilizadas por todos os outros *layouts*, quando a velocidade e a posição são atualizadas, até que se encontre outro *gbest* melhor.

Neste algoritmo, o método enxame de partícula obteve melhores resultados em todas as instâncias testadas, comparando-se com os resultados obtidos por Anjos e Vannelli (2002). Para visualização concreta de um *layout* industrial, a partir dos resultados obtidos, foram gerados *layouts* retangulares.

Em vista a todas as considerações, o uso do método enxame de partículas é indicado na otimização do problema de *layout*. No entanto, por ser um método relativamente recente, pode ser mais explorado e, talvez, aperfeiçoado.

REFERÊNCIAS

ABINO, M.A. Optimal design of power-system stabilizers using particle swarm optimization. *IEEE Trans. Energy Conv.*, vol. 17, n. 3, sept. 2002, p. 406-413.

ALBRECHT, C.H. *Algoritmos evolutivos aplicados a síntese e otimização de sistemas de ancoragem.* COPPE/Ufrj, 2004.

ALVARENGA, A.G. de; NEGREIROS-GOMES, F.J.; MESTRIA, M. Metaheuristic methods for a class of the facility layout problem. *Journal of Intelligent Manufacturing*, Vitória, n.11, 2000, p. 421-430.

ANDRADE, Eduardo Leopoldino de. *Introdução à Pesquisa Operacional*. 2.ed. Rio de Janeiro: LTC, 1998.

ANGELINE, P.J. A historical perspective on the evolution of executable structures. *Informaticae*, v. 36, 1998, p. 179-195.

ANJOS, M.F.; VANNELLI, A. An attractor-repeller approach to floorplanning. *Mathematical Methods of Operations Research*, n. 1, 2002, p. 3-27.

ARMOUR, G.C.; BUFFA, E.S. A heuristic algorithm and simulation approach to the relative location of facilities. *Management Science*, n.9, 1963, p. 294-309.

AZADIVAR, F.; WANG, J. Facility layout optimization using simulation and genetic algorithms. *International Journal of Production Research*, v. 38, n. 17, 2000, p. 4369-4383.

BAZARAA, M.S.; ELSHAFEI, A.N. An exact branch and bound procedure for quadratic assignment problems. *Naval Research Logistics Quarterly*, n. 26, 1979, p. 109-121.

BURKARD, R.E.; RENDL, F. A Thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems. *European Journal of Operational Research*, n. 17, 1984, p. 169-174.

CASTILLO, I.; SIM, T. *A Spring-Embedding Approach for the Facility Layout Problem*. University of Alberta School of Business, Edmonton, Alberta, Canada, 2003.

CHATTERJEE, A.; SIARRY, P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers & OR*, n. 33, 2006, p. 859-871.

CHIANG, W. C. Visual facility layout design system, *International Journal Production Research*, v. 39, n. 9, 2001, p. 1811-1836.

CLERC, M.; KENNEDY, J. *The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Somplex space*. IEEE Transaction on Evolutionary Computation, v. 6, 2002, p. 58-73.

CORTES, M.B.S. *Algoritmos Genéticos em Problemas de Programação Não Linear Contínua*. (Tese de Doutorado). UFSC, Florianópolis, 1996.

DICKEY, J.W.; HOPKINS, J.W. Campus building arrangement using topaz. *Transportation Research*, n. 6, 1972, p. 59-68.

DORIGO, M. *Optimization, Learning and Natural Algorithms*. Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

DREZNER, Z. DISCON: a new method for the layout problem. *Operations Research*, n. 28, 1980, p. 1375-1384.

_____. A heuristic procedure for the layout of a large number of facilities. *Management Science*, n. 33, 1987, p. 907-915.

EBERHART, R.C.; SIMPSON, P.K.; DOBBINS, R.W. *Computational Intelligence PC Tools.* Boston, MA: Academic Press Professional, 1996.

EBERHART, R.C.; HU, X. Human tremor analysis using particle swarm optimization. Proc. Congress on Evolutionary Computation 1999, Washington, DC, Piscataway, NJ: IEEE Service Center, 1999, p. 1927-1930.

EBERHART, R.C.; SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. Proceedings *of Congress on Evolutionary Computation*, San Diego, 2000, p. 84-88.

EBERHART R C, SHI, Y. Particle swarm optimization: development, applications and resources. In: *Proceedings of Congress on Evolutionary Computation*, Seoul, Korea, 2001, p.81-86.

ELSHAFEI, A.N. Hospital layout as a quadratic assignment problem. *Operations Research Quarterly*, n. 28, 1977, p. 167-179.

FEO, T.; RESENDE, M. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, n. 6, 1995, p. 109-133.

FOULDS, L.R.; ROBINSON, D.F. Graph theoretic heuristics for the plant layout problem. *International Journal of Production Research*, v. 16, n. 1, 1978, p. 27-37.

FURTADO, J.C.; LORENA, L.A.N. Otimização em Problemas de Leiaute. In: XX Congresso Nacional de Matemática Aplicada e Computacional e 2ª Oficina Nacional de PCE, 1997,

Gramado - RS. *Anais da 2ª Oficina Nacional de Problemas de Corte e Empacotamento*, 1997a, p. 129-146.

_____. Otimização de leiaute usando busca tabu. *Gestão & Produção*, vol.4, n.1, 1997b.

GAITHER**,** N.; FRAZIER**,** G. *Administração da produção e operações*. 8.ed. São Paulo: Pioneira, Thomson Learning, Inc., 2001.

GLOVER, F. Tabu Search – part I, *ORSA Journal on Computing*, vol I, 1989a, p. 190-206.

_____. Tabu Search – part II, *ORSA Journal on Computing*, vol II, 1989b, p. 4-32.

GOMES, A.; CREÃO, D. Enxame de Partículas Evolucionários. In: MIRANDA, Vladimiro. *Computação Evolucionária Fenotípica*. Faculdade de Engenharia do Porto, 2004.

GOMORY, R. E. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, n. 64**,** 1958.

GREFENSTETE, J.J. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, n. 16, 1986, p. 122-128.

GUIMARÃES, Antonio Cesar Ferreira. *Enxame de partículas como ferramenta de otimização em problemas complexos de Engenharia Nuclear*. Tese (Doutorado em Engenharia Nuclear) - Universidade Federal do Rio de Janeiro, 2005.

HERAGU, S.S.; ALFA, A.S. Experimental Analysis of Simulated Annealing based Algorithms for the Layout Problem. *European Journal of Operational Research*, n. 57, 1992, p. 190-202.

HERAGU, S.S.; KUSIAK, A. Efficient Models for the Facility Layout Problem. *European Journal of Operational Research*, n. 53, 1991, p. 1-13.

HOLLAND, J.H. *Adaptation in natural and artificial systems*. University of Michigan Press: Ann Harbor, MI, 1975.

IBARAKI, T. Combinatorial Optimization Problems and their Complexity. In: Enumerative Approaches to Combinatorial Optimization - Part I. *Annals of Operations Research*, v. 10. 1988.

KASSABALIDIS, I.N.; EL-SHARKAWI, M.A.; MARKS, R.J.; MOULIN, L.S.; SILVA, A.P.A. da. Dynamic security border identification using enhanced particle swarm optimization. *IEEE Trans. Power Syst.*, vol. 17, aug. 2002, p. 723-729.

KENNEDY, J. *The behavior of particles*. In: Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on evolutionary programming, San Diego, CA, 1998, p. 581-589.

KENNEDY, J.; EBERHART, R. C. *Particle Swarm Optimization*. In: The 1995 IEEE International Conference on Neural Networks, Perth, Australia, 1995, p. 1942–1948.

KENNEDY, J.; EBERHART, R.C.; SHI, Y. *Swarm intelligence*. San Francisco: Morgan Kaufmann Publishers, 2001.

KIRKPATRICK, S.; GELLAT, C. D.; VECHI, M. P. Optimization by Simulated Annealing. *Science*, n. 220, 1983, p. 671-680.

KOAKUTSU, S.; HIRATA, H. *Genetic Simulated Annealing for Floorplan Design*. Chiba University, Japan, 1992.

KOOPMANS, T.C.; BECKMAN, M. Assignment Problems and the Location of Economic Activities. Econometrica, n. 25, 1957, p. 53-76.

LEE, R.C.; MOORE, J.M. CORELAP: Computerized Relationship Layout Planning. *Industrial Engineering*, n. 18, 1967, p. 195-200.

LIM, A.; LIN, J.; XIAO, F. Particle Swarm Optimization and Hill Climbing to Solve the Bandwidth Minimization Problem. *MIC2003: The Fifth Metaheuristics International Conference*, Kyoto, Japan, aug. 2003, p. 25-28.

LOIOLA, E.M.; ABREU, N.M.M. de; NETTO, P.O.B.. Uma revisão das abordagens do Problema Quadrático de Alocação. *Pesquisa Operacional*, v. 24, n. 1, jan-abr. 2004, p. 73-109.

MARQUES, S.R.A. *Projeto de layout industrial no contexto just in time auxiliado por computador*. (Dissertação de Mestrado) UFSC, Florianópolis, 1993.

MARTINS, V.C.; COELHO, L. dos S.; CÂNDIDO, M.A.B.; PACHECO, R.F. Otimização de layouts industriais com base em Busca Tabu. *Gestão & Produção*, vol.10, n.1, São Carlos, apr. 2003.

MAVRIDOU, T.D.; PARDALOS, P.M. Simulated Annealing and Genetic Algorithms for the Facility Layout Problem: a survey. *Computational Optimization and Applications*, n. 7, 1997, p. 111-126.

MLADENOVIC, N. A variable neighborhood algorithm: a new metaheuristic for combinatorial optimization. In: *Abstracts of Papers at Optimization Days*, n. 112, 1995.

MLADENOVIC, N.; HANSEN P. Variable neighbourhood search. *Computers and Operations Research*, n. 24, 1997, p. 1097-1100.

MONTREUIL, B. *A modeling framework for integrating layout design and flow network design.* Proceeding of the Material Handling Research Colloquium, Hebron, KY, 1990, p. 43-58.

MONTREUIL, B.; VENKATADRI, U.; RATLIFF, H. D. Generating a layout from a design skeleton. *IIE Transactions*, v. 25, n. 1, 1993, p. 3-15.

MOURA, D.A. de; MARTINS, L.A.P; DUARTE, M.L. de A. Aplicação das Técnicas e Ferramentas da Disciplina Administração da Produção e Operações, Visando Aumento da Qualidade e Produtividade num Centro de Distribuição Domiciliar dos Correios. *Administração On Line*, v. 3, n. 2, abril/maio/junho, 2002.

NUGENT, C.E.; VOLLMANN, T.E.; RUML, J. An Experimental Comparision of Techniques for the Assigment of Facilities to Locations. *Operations Research*, n. 16, 1968, p. 150-173.

PARK, Jong-Bae; LEE, Ki-Song; SHIN, Joong-Rin; LEE, Kwang Y. A Particle Swarm Optimization for Economic dispatch with nonsmooth cost functions. *IEEE Transactions on Power Systems*, vol. 20, n. 1, February, 2005.

PARSOPOULO, K.E; VRAHATIS, M.N, *Particle Swarm Optimization Method in Multiobjective Problems*. In: Proceedings of the 2002, University of Patras, Greece, 2002.

PRADO, J.R.do; SARAMAGO, S.F.P. Otimização por Colônia de Partículas. *FAMAT em Revista*, n. 04, abr. 2005.

RATNAWEERA, A; HALGAMUGE, S.K.; WATSON, H.C. *Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients*. IEEE Transactions on Evolutionary Computation, v. 8, 2004, p. 240-255.

SAHNI, S.; GONZALEZ, T. P-complete approximation problems. *Journal of the Association of Computing Machinery*, n. 23, 1976, p. 555-565.

SALOMÃO, Silvely Nogueira de Almeida. *Métodos de Geração de Colunas para Problemas de Atribuição*. São José dos Campos: INPE, 2005.

SANTOS, Antônio Raimundo dos. *Metodologia Científica*: a construção do conhecimento. 3.ed. Rio de Janeiro: DP&A Editora, 2000.

SCHAFFER, J.D.; CARUANA, R.A.; ESHELMAN, L.J.; DAS, R. A study of controle parameters affecting online performance of Genetic Algorithms for function optimization. Proceedings on the Third International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, 1989, p. 51-60.

SCRIABIN, M.; VERGIN, R.C. A cluster-analytic approach to facility *layout*. *Management Science*, n. 31, 1985, p. 33-49.

SEEHOF, J.M.; EVANS, W.O. Automated layout design program. *Industrial Engineering*, n. 18, 1967, p. 690-695.

SHI, Y.; EBERHART, R.C. A modified particle swarm optimizer. *Proceedings of the IEEE International Conference on Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1998, p. 69-73.

TAM, K.Y. Genetic algorithms, function optimization and facility layout design. *European Journal of Operational Research*, n. 63, 1992, p. 322-346.

TAM, K.Y.; LI, S.G. A hierarchical approach to the facility layout problem. *International Journal of Production Research*, n. 29, 1991, p. 165-184.

TANDON, V. Closing the gap between CAD/CAM and optimized CNC en milling. Master's thesis, Purdue School of Engineering and Technology, Indiana University Purdue University Indianapolis, 2000.

TATE, D. M.; SMITH, A. E. A genetic approach to the quadratic assignment problem. *Computers and Operations Research,* v. 32, 1995, p. 73-83

TAVARES, José Antônio dos Reis. *Geração de Configurações de Sistemas Industriais com o Recurso à Tecnologia das Restrições e Computação Evolucionária*. Tese de Doutorado. Universidade do Moinho, Porto, 2000.

TOMPIKINS, J. A.; REED Jr., R. An applied model for the facilities design problem. *International Journal of Production Research,* v. 14, n. 5, 1976, p. 583-595.

VAN CAMP, D.J.; CARTER, M.W.; VANNELLI, A. A nonlinear optimization approach for solving Facility *Layout* Problems. European Journal of Operational Research, n. 57, 1991, p. 174-189.

VOLLMAN, T.E.; BUFFA, E.S. Facilities layout problem in perspective. *Management Science*, n. 12, 1966, p. 450-468.

YOSHIDA, H.; KAWATA, K.; FUKUYAMA, Y.; NAKANISHI, Y. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Trans. Power Syst.*, vol. 15, nov. 2000, p. 1232-1239.