



Universidade do Porto

Faculdade de Engenharia

FEUP

Nuno Miguel Duarte Sequeira André

PARTICLE SWARM OPTIMIZATION

Optimization and decision support techniques
PDEEC 2008

Introdução

O algoritmo PSO é um algoritmo de otimização estocástico e baseado em populações

É uma espécie de inteligência colectiva de um enxame baseada em princípios psico-sociológicos

Pode ser usado para demonstrar comportamentos sociais ou aplicações de engenharia

Foi inicialmente descrito em 1995 por James Kennedy e Russell C. Eberhart mas evoluiu muito desde então

Introdução

A influência e aprendizagem social permitem ao indivíduo ter consistência cognitiva

Resolvemos problemas discutindo-os

À medida que interagimos, as nossas crenças, atitudes e comportamentos mudam

Podemos encarar estas mudanças como movimentações num espaço sócio-cognitivo

Introdução

O PSO simula este tipo de otimização social

Para um determinado problema TSP temos de determinar o custo de várias soluções ou caminhos

É definida ainda uma estrutura de comunicação

A população é definida como soluções aleatórias para o problema em que todos os indivíduos são candidatos à solução óptima

São conhecidos como partículas, daí o nome
Particle Swarm

Introdução

É então iniciado o processo que melhora as soluções candidatas

As partículas avaliam o custo da solução candidata e lembram-se qual a melhor solução onde estiveram

Esta solução é conhecida como o mínimo da partícula ou mínimo local

Esta informação também é dada aos seus vizinhos, sabendo estes onde se obtiveram os melhores sucessos

Introdução

O mínimo global é também conhecido de todas as partículas

Os movimentos através do espaço de pesquisa são guiados por estas informações

A população geralmente converge numa solução melhor do que se usássemos os mesmo métodos mas sem a comunicação

Introdução

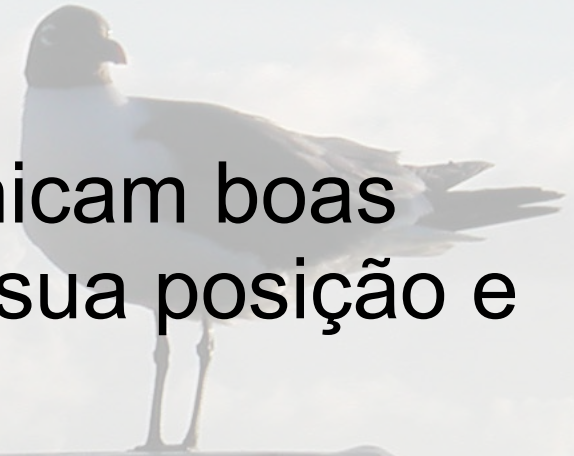
O enxame de partículas é geralmente modelizado como partículas num espaço multidimensional que têm uma posição e velocidade

Estas partículas deslocam-se neste espaço (que é o espaço das soluções) tendo uma posição e velocidade

Têm ainda duas capacidades decisórias: a memória da sua melhor posição e o conhecimento do mínimo global ou do mínimo local das partículas vizinhas

Introdução

Os membros do enxame comunicam boas posições entre si e ajustam a sua posição e velocidade



Introdução

O sistema é baseado numa equação que determina as alterações que as partículas fazem mediante as informações que obtêm

$$v_{i,j} = c_1 v_1 + c_2 (\textit{mínimo partícula}_j - x_{i,j}) + c_3 (\textit{mínimo global}_j - x_{i,j})$$
$$x_{i,j} = x_{i,j} + v_{i,j}$$

cada constante c_1, c_2, c_3 representa a inércia de cada parâmetro

depois da velocidade calculada muda-se a variável para uma nova posição

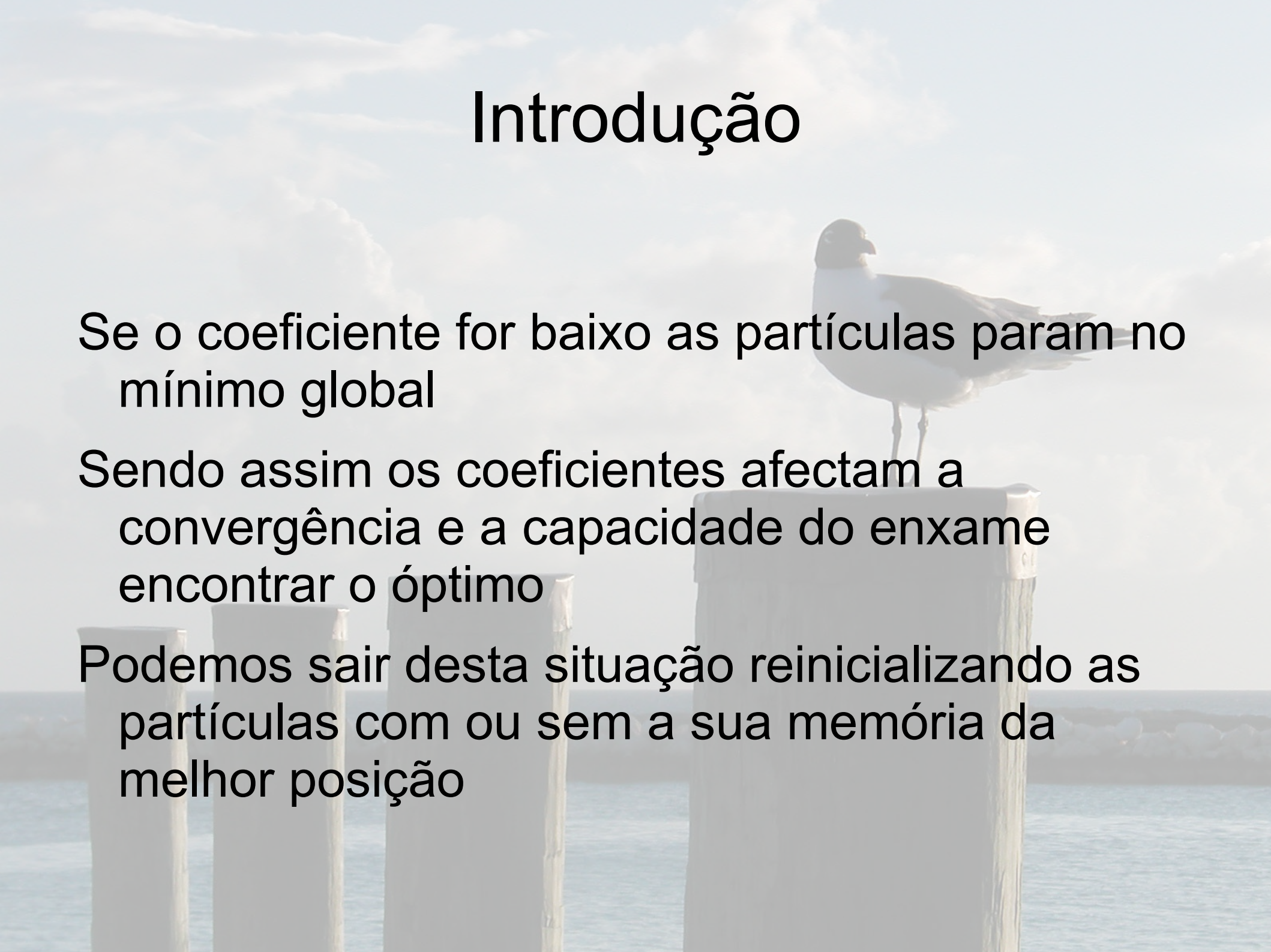
Introdução

À medida que o enxame faz iterações o custo global desce

As partículas podem ser influenciadas por este resultado e aproximarem-se dele nunca melhorando depois de lá chegarem independentemente do número de iterações

Caso as partículas se movam próximo do mínimo global sem pesquisarem o resto do espaço estamos perante o fenómeno da convergência

Introdução

A seagull is perched on a wooden post, looking towards the left. The background shows a bright, cloudy sky and a body of water. The text is overlaid on the left side of the image.

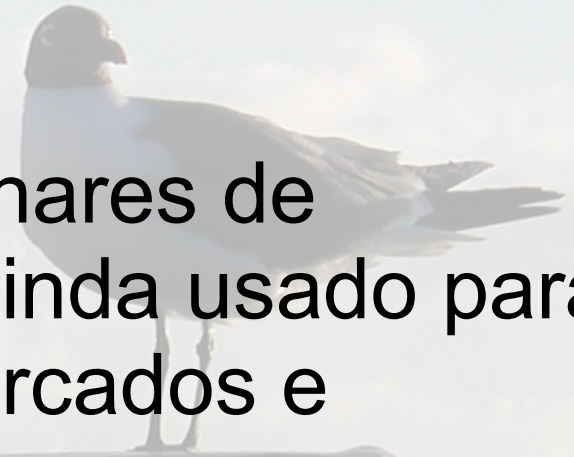
Se o coeficiente for baixo as partículas param no mínimo global

Sendo assim os coeficientes afectam a convergência e a capacidade do enxame encontrar o óptimo

Podemos sair desta situação reiniciando as partículas com ou sem a sua memória da melhor posição

Introdução

Este algoritmo foi usado em milhares de aplicações de engenharia, é ainda usado para compor musica, modelizar mercados e organizações



Exemplo

Uma forma mais simples de explicação pode ser um bando de gaivotas telepáticas à procura da melhor fonte de comida

Todas começam num mapa distribuídas aleatoriamente

No início antes de se moverem têm velocidade zero mas sabem qual a gaivota mais próximo da solução óptima

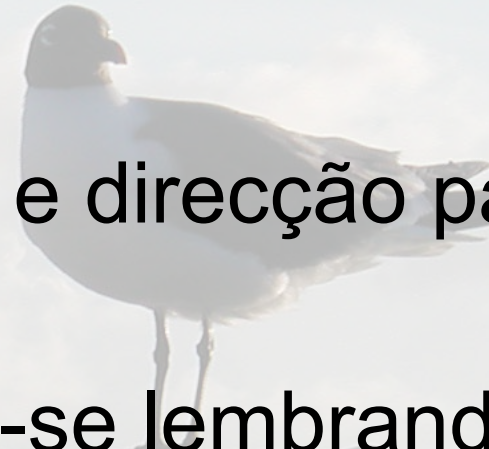


Exemplo

Alteram então a sua velocidade e direcção para irem nesse sentido

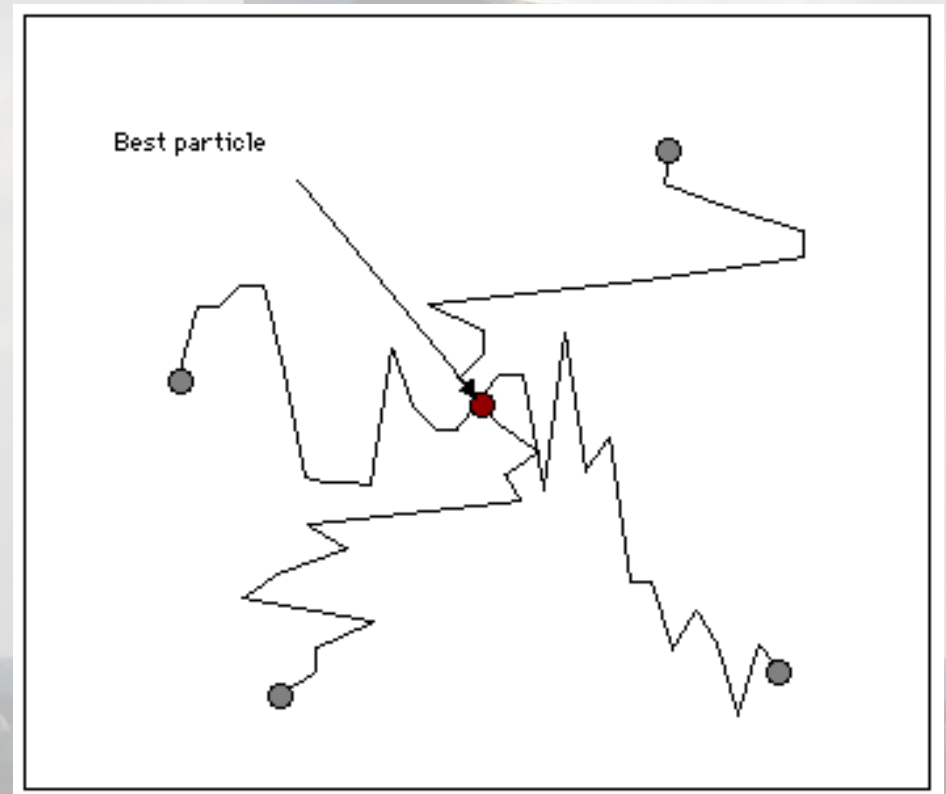
À medida que vão andando vão-se lembrando da sua melhor posição

Usam a sua velocidade actual, a sua melhor posição e a melhor posição global para alterarem a sua velocidade e logo a sua posição para se dirigirem para a posição óptima



Exemplo

Como iniciaram em posições aleatórias no mapa ao dirigirem-se todas para o mesmo ponto inevitavelmente vão passar pelo ponto óptimo, alterando ao melhor global e fazendo com que todas consigam convergir nesse ponto



Implementação



A implementação começa por ser um pouco complicada, pois estamos perante um problema discreto e a equação lidaria melhor com variáveis contínuas

Encontramos problemas como subtrair uma posição a outra

Com a solução destes problemas criam-se os blocos constituintes básicos do algoritmo

Implementação

A seagull is perched on a wooden post, looking towards the left. The background is a bright, cloudy sky. The text is overlaid on the image.

A parte inicial do algoritmo tem de ser inevitavelmente a geração de partículas, para tal usa-se um vector bidimensional sendo cada linha uma partícula diferente e tendo tantas colunas como nós do problema

Este vector contém as posições das partículas

Inicialmente são gerados todos os caminhos por ordem e depois são trocados aleatoriamente para garantir que sejam diferentes

Implementação

De seguida temos de calcular o custo de todos os caminhos para ver qual o que tem o menor custo e actualizar o melhor caminho global, são também actualizados os melhores caminhos de sempre de cada partícula, usando-se para isso um vector bidimensional com a mesma dimensão do vector das posições

Implementação

Calcular as velocidades das partículas é a parte mais complexa e na qual se pode usar a maior criatividade para transformar algo que é contínuo em algo discreto

As velocidades são guardadas num vector tridimensional que na verdade é equivalente a dois vectores com as mesmas linhas do vector das posições, mas optei por usar um tridimensional em vez de dois separados, um guarda a posição inicial de um nó e o outro a posição final do nó

Implementação

Sendo assim a velocidade é expressa pelo número de trocas que são feitas num caminho fazendo com que a partícula se desloque no espaço das soluções

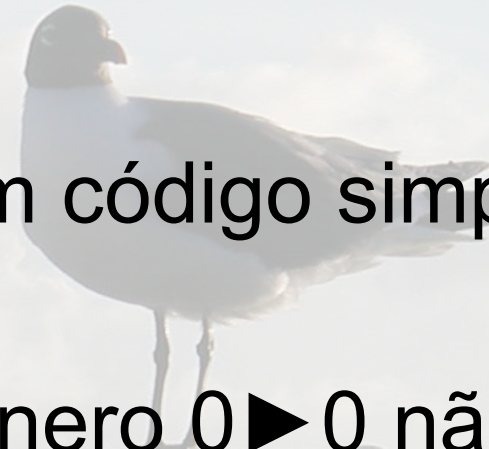
Se uma partícula tiver a velocidade $0 \blacktriangleright 1$, vamos trocar os nós que estão na posição 0 com a 1, esta é a velocidade mais baixa que pode ter

Se a velocidade fosse $0 \blacktriangleright 1, 4 \blacktriangleright 2, 1 \blacktriangleright 7, 2 \blacktriangleright 3$ já seria uma velocidade superior

Implementação

O uso de trocas permite usar um código simples e rápido

Quando temos uma troca do género $0 \blacktriangleright 0$ não há movimentação



Implementação

As constantes são usadas para determinar quanto da velocidade inicial vamos manter

Se c_1 fosse 2 e se a velocidade da partícula fosse: $0 \blacktriangleright 1$, $4 \blacktriangleright 2$, $1 \blacktriangleright 7$, $2 \blacktriangleright 3$

Iria-se escolher aleatoriamente duas trocas para serem guardadas que depois se iriam acrescentar às outras velocidades de c_2 e c_3

$$v_{i,j} = c_1 v_1 + c_2 (\text{mínimo partícula}_j - x_{i,j}) + c_3 (\text{mínimo global}_j - x_{i,j})$$

$$x_{i,j} = x_{i,j} + v_{i,j}$$

Implementação

Para calcular a subtracção de duas posições (em $c2$ e $c3$) basta escolher aleatoriamente uma posição no caminho mínimo da partícula (ex 5), de seguida pesquisamos a posição actual e determinamos onde se encontra o nó que está na posição 5 do mínimo da partícula, (ex 7)

Guardamos assim a velocidade $5 \blacktriangleright 7$, repetimos tantas vezes como $c2$

Para $c3$ será idêntico

Implementação

$c2(\text{Melhor Posição} - \text{Posição actual})$

[3 6 8 2 5 **1** 0 9 4 7] Melhor posição

[9 5 7 4 2 3 6 **1** 0 8] Posição actual

5 ► 7 Resultado

Repetir tantas vezes quanto $c2$

Implementação

O problema com este código até agora é que ele vai convergir relativamente rápido mesmo para valores baixos de c_1 , c_2 e c_3

Experimentei várias maneiras de reiniciar os nós:

- Reiniciar a partícula se a sua posição for igual à melhor global

- Reiniciar a partícula se a sua melhor posição for igual à melhor global

Nos dois casos testei reiniciar a partícula só na posição ou na posição e na sua memória

Implementação

Finalmente optei por reiniciar só a posição da partícula quando esta é igual à melhor global evitando assim a convergência das posições das partículas e das suas memórias que também convergiam

Reiniciar a memória levava muito tempo e não trazia ganhos significativos

Implementação

Resta agora determinar quais os melhores valores para c_1 , c_2 e c_3

A maioria da literatura recomenda que estes sejam dois

Realizei várias iterações tentando todas as combinações entre 1 e 3

Combinei também o n° de iterações e n° de partículas para tentar determinar a melhor configuração

Resultados

TSP 53 Nós (OPT 426)											
Iterações			100			1000			10000		
Partículas			100	1000	10000	100	1000	10000	100	1000	10000
C1	C2	C3									
1	1	1	1241	1106	1120	689	635	504	541	548	513
1	1	2	1160	1041	887	659	583	555	547	547	511
1	1	3	1016	880	717	687	553	515	610	535	548
1	2	1	1125	1150	1121	636	557	519	483	515	463
1	2	2	1108	999	969	679	579	526	505	558	484
1	2	3	1078	871	826	683	537	491	592	520	501
1	3	1	1219	1109	1084	635	568	534	577	518	502
1	3	2	1150	990	943	659	535	499	617	570	490
1	3	3	1135	911	795	597	601	513	521	505	549
2	1	1	1265	1204	1158	711	617	581	573	516	513
2	1	2	1167	952	897	630	606	507	521	510	492
2	1	3	939	919	809	610	592	518	565	504	546
2	2	1	1207	1147	1080	719	607	555	589	535	479
2	2	2	1151	1048	923	662	601	529	533	509	520
2	2	3	1017	972	800	662	547	568	532	550	521
2	3	1	1214	1118	1123	706	566	511	555	553	536
2	3	2	1203	1030	948	598	557	495	527	495	518
2	3	3	991	829	830	546	588	537	556	520	491
3	1	1	1260	1211	1084	739	604	544	498	513	517
3	1	2	1089	1018	872	702	542	558	521	525	504
3	1	3	996	884	805	614	583	503	574	571	515
3	2	1	1241	1137	1106	746	595	539	560	509	500
3	2	2	1098	999	962	674	527	589	555	553	487
3	2	3	944	901	815	684	588	559	578	558	523
3	3	1	1241	1104	1065	688	548	488	579	513	497
3	3	2	1140	1091	935	632	509	534	595	590	551
3	3	3	1060	975	763	553	598	515	539	513	505
Mínimo			939	829	717	546	509	488	483	495	463
Mínimo Iter.			717			488			463		
Tempo			4m 26s			42m 51s			7h 14m 51s		

Res.

TSP 1002 Nós (OPT 259045)											
Iterações			100			1000			10000		
Partículas			100	1000	10000	100	1000	10000	100	1000	10000
C1	C2	C3									
1	1	1	6175899	6123253	Erro Memória	6128209	6017094	Erro Memória	5629764	5363907	Erro Memória
1	1	2	6154663	6085222		6066798	5972258		4965325	4799287	
1	1	3	6089166	6135797		5929106	5913730		4803782	4511442	
1	2	1	6245114	6142751		6075874	5981821		5525547	5481598	
1	2	2	6163874	6107420		6043868	5932757		5030558	4713859	
1	2	3	6165218	6118187		6008205	5842482		4923936	4329753	
1	3	1	6227502	6090446		6072899	5982083		5426596	5419755	
1	3	2	6171850	6127526		5990620	5966639		4853841	4605911	
1	3	3	6155283	6121910		5972186	5862587		4868095	4346273	
2	1	1	6198186	6150074		6128924	6037798		5608551	5506772	
2	1	2	6166439	6121297		6048891	6029267		5007478	4744613	
2	1	3	6139590	6060715		5996560	5932662		4920405	4471671	
2	2	1	6212508	6120812		6131141	6065865		5591086	5463500	
2	2	2	6191177	6139550		6035546	5979508		4977937	4722023	
2	2	3	6172703	6097076		5984093	5971803		4839815	4349140	
2	3	1	6191182	6092355		6040482	6022789		5451156	5503787	
2	3	2	6172376	6133253		6040294	5927738		4733748	4876062	
2	3	3	6161126	6096366		5976086	5866005		4669472	4414657	
3	1	1	6244312	6096390		6135172	6039698		5657212	5345150	
3	1	2	6178486	6125104		6045450	5985800		5272960	4903241	
3	1	3	6182656	6120583		6024293	5929878		4935007	4475509	
3	2	1	6267255	6136333		6096571	6031307		5578759	5526622	
3	2	2	6111278	6123180	6011227	5955976	5027061	4984349			
3	2	3	6211329	6063933	5999563	5927333	4700147	4365646			
3	3	1	6193280	6118864	6038512	6011490	5556012	5566436			
3	3	2	6181652	6112991	6047337	5944500	4921160	4981338			
3	3	3	6201284	6100010	5953123	5918473	4721977	4571706			
Mínimo			6089166	6060715		5929106	5842482		4669472	4329753	
Mínimo Iter.			6060715			5842482			4329753		
Tempo			46m 18s			2h 2m 5s			18h 16m 12s		

Conclusão

Tendo em conta que o PSO não é de todo o ideal para problemas do género TSP este comportou-se bastante bem, conseguindo resultados algo próximos do óptimo, para 53 nós em muito menos tempo que o VNS

O facto de o PSO não ser ideal para TSP prova-se no caso dos 1002 nós pois necessitaríamos de muito mais partículas do que a memória permite e de muitas iterações, visto que o resultado foi aproximadamente 6M, muito longe do resultado de qualquer heurística simples

Conclusão

Sendo assim o grande problema do PSO é o número de partículas/iterações necessárias para se conseguir uma boa solução

Como temos de chegar a uma boa solução aleatória para as partículas tentarem melhorar, se não chegarmos a uma solução aceitável logo no início as partículas dificilmente a encontrarão

Como se nota variar c_1 , c_2 e c_3 não tem grandes efeitos na solução, pode-se usar todas as constantes com o valor 2 como recomendado

Conclusão

Já o número de iterações melhora as soluções obtidas muito mais do que o número de partículas, o que se explica com o facto de reiniciarmos as partículas muito mais vezes com muitas iterações do que tendo muitas partículas e poucas iterações, como elas são reiniciadas tendo uma memória que tem mais tempo para evoluir consegue-se melhores resultados

Verifiquei ainda uma das vantagens do PSO: a versatilidade, visto que não necessitei de afinar parâmetros

Bibliografia

Xiaohui Hu, “Particle Swarm Optimization”,
<http://www.swarmintelligence.org/index.php>, 2006
(Visitada em 03/02/2008)

http://en.wikipedia.org/wiki/Particle_Swarm_Optimization
(Visitada em 03/02/2008)

Maurice Clerc, “Discrete Particle Swarm Optimization
Illustrated by the Traveling Salesman Problem”,
http://clerc.maurice.free.fr/pso/pso_tsp/Discrete_PSO_TSP,
2000 (Visitada em 03/02/2008)

Maurice Clerc, “Discrete Particle Swarm Optimization: A
Fuzzy Combinatorial Black Box”,
http://clerc.maurice.free.fr/pso/Fuzzy_Discrete_PSO/Fuzzy,
2000 (Visitada em 03/02/2008)