Faculdade de Engenharia da Universidade do Porto

Organização da rede de abastecimento da Green-NET

Sistemas de Apoio à Decisão

Gonçalo Resende, up201707219@edu.fe.up.pt

Nuno Minhoto, up201604509@edu.fe.up.pt

Pedro Aidos, up201706842@edu.fe.up.pt

Índice

1.Introdução

Introdução e descrição do problema

O problema a abordar neste trabalho consiste no desenvolvimento de um SAD para uma cooperativa, Green-NET. Esta instituição possui uma rede de vários armazéns e várias lojas, as quais pretende abastecer diariamente com pequenos “cabazes” de produtos variados.

Cada loja comunica à organização a sua necessidade em número de cabazes, sendo que cada armazém tem um número limitado de cabazes que podem fornecer em cada dia. A distribuição dos cabazes pelas lojas é feita por um veículo, tendo cada armazém apenas um veículo disponível, que apenas segue uma rota. Cada loja só pode ser abastecida por um armazém e a procura total de todas as lojas que sejam servidas por um armazém tem de ser menor do que a capacidade do mesmo. A empresa tem custos diretamente proporcionais à distância que cada veículo percorre ao sair do armazém. Pretende-se que esses custos sejam reduzidos ao máximo, ao mesmo tempo que se minimiza a distância entre cada loja e o armazém que a serve.

Justificação do recurso a um SAD

Para cumprir com os requisitos apresentados acima, um SAD é um sistema bastante útil, uma vez que se trata de um problema de otimização não linear. Este permite o cálculo de uma solução viável de forma rápida e dinâmica. Isto torna-se muito importante porque a empresa também tem como principal objetivo a diminuição da distância entre os armazéns e os seus clientes logo torna-se útil a utilização de um sistema onde o decisor pode definir pesos para os diferentes critérios. Espera-se que a rota seja definida de uma forma bastante otimizada de maneira que os recursos usados na distribuição sejam o mínimo possível. É também importante mencionar que num programa desta ordem a sua capacidade de chegar a uma solução melhor está dependente dos recursos disponíveis, isto é, se tivermos mais tempo disponível para deixar o programa correr a solução virá a tornar-se melhor, no entanto apresenta sempre uma solução no tempo que tiver disponível.

2.O SAD

Modelo matemático do problema

Para tornar o problema mais percetível do ponto de vista de implementação de um SAD, das heurísticas a implementar, é importante saber as variáveis associadas, a função objetivo e as restrições associadas. Assim sendo ficamos com o seguinte modelo.

Dados:

j – Armazém

i – Loja

c_j – Capacidade do armazém j

d_i – Procura da loja i

Variáveis de decisão:

X_{ij} – 1 se a loja i vai ser abastecida pelo armazém j; 0 caso contrário

Variáveis auxiliares:

S – Distancia percorrida na rota

Função objetivo:

Minimizar os custos

Minimizar distancia dos armazéns aos clientes

Restrições:

- só abastece as lojas se o armazém tiver capacidade para isso $\sum_i di * xij \leq cj, \forall j$
- só é abastecido por um armazém $\sum_j xij \leq 1, \forall i$

Arquitetura do sistema

O SAD é um algoritmo que vai ajudar o utilizador a definir as rotas, neste caso, e sendo apenas um mecanismo de ajuda tem de haver uma forte interação com o utilizador. Pode haver a necessidade de criar um novo armazém, ou fechar, assim como pode existir contratos com novas lojas, ou algumas que cancelem o contrato existente. Para responder a esta instabilidade é necessário criar um algoritmo dinâmico capaz de sofrer alterações.

Para responder a esta necessidade o nosso programa é capaz de ler os armazéns e lojas e respetivas posições de um ficheiro txt, que posteriormente pode ser adaptado para uma base de dados, e podem ser acrescentados ou removidos através da interface gráfica a que o utilizador vai ter acesso.

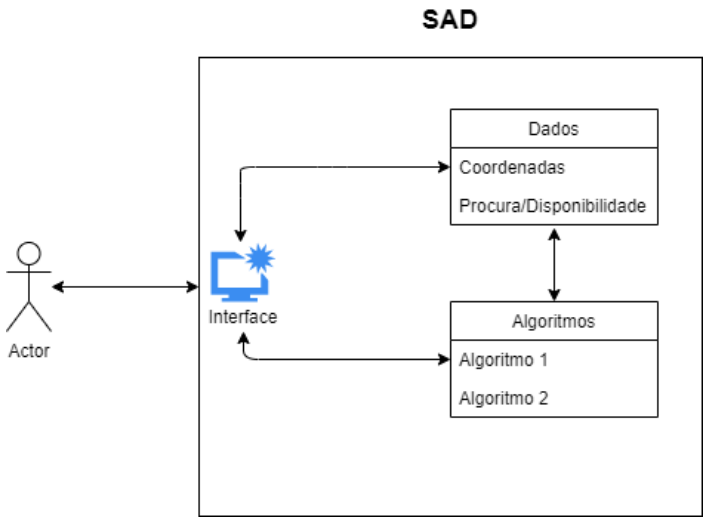

Figura 1- Esquema da arquitetura do sistema

A interface permite também a valorização de diferentes critérios para priorizar o que o utilizador considerar prioritário visto que se trata de um software de ajuda na decisão de problemas multiobjectivo.

Nas seguintes figuras fica uma ideia de uma possível interface dando enfase às questões abordadas acima.

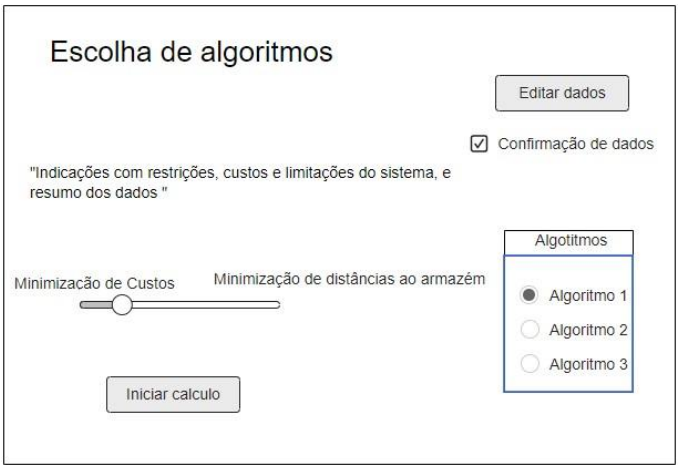

Figura 2- Alterar o objetivo

Figura 3- Introduzir loja ou/e armazém

Figura 4- Apresentação dos resultados

A resolução deste problema baseia-se na implementação de heurísticas, primeiro construtivas e depois de melhoramento. Podemos ter como base problemas bastante estudados nesta área, os problemas de mochila porque queremos pegar em lojas e distribuir por armazéns e o problema do caixeiro-viajante, que consiste em percorrer determinados sítios com a menor distancia possível, isto é o que nos interessa também na definição da rota que cada distribuir vai percorrer. Com base nestes modelos criamos duas heurísticas diferentes, uma que dá prioridade ao objetivo de diminuir a distância total e outra para minimizar a distância de cada loja ao respetivo armazém.

Planeamento do projeto

Sendo este um projeto complexo há necessidade de dividir em tarefas mais simples e ordená-las para que se consiga fazer uma melhor gestão do projeto. Todas as principais tarefas já foram mencionadas ao longo deste relatório. Começar por perceber o problema e fazer o seu modelo matemático, onde se torna bastante simples saber qual é o objetivo do SAD a implementar; criar uma possível interface de maneira a que o cliente e quem vai interagir com o nosso programa possa perceber como esta interação é feita e também se está de acordo com as suas espectativas quando decidiram recorrer a um software deste tipo; definir as heurísticas construtivas que vão levar a cumprir os objetivos previamente definidos, e posteriormente definir uma heurística de melhoramento que vai ajudar a obter um melhor resultado. Com as heurísticas definidas passa-se para a parte de implementação, também pela mesma ordem, começar pelas heurísticas construtivas para obtermos uma primeira solução viável, e depois com a implementação de uma meta heurística tornar essa solução o melhor possível.

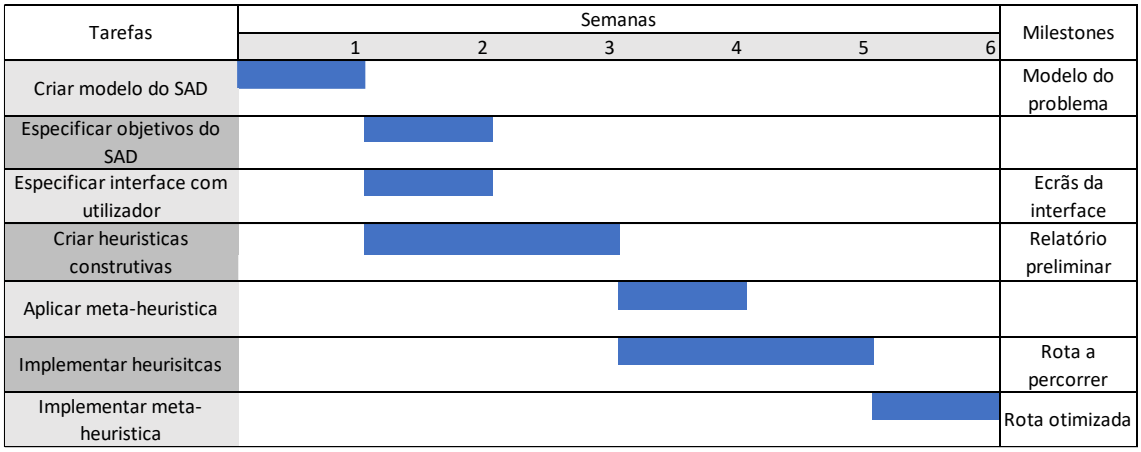

Figura 5- Gantt planeamento do projeto

3.Algoritmos

Heurísticas

As heurísticas construtivas, são as primeiras a ser definidas e implementadas pois são estas que vão criar a primeira rota a partir da qual depois se podem vir a fazer melhoramentos. Sendo este um problema multiobjectivo justifica-se o recurso a duas heurísticas construtivas diferentes.

Heurística de diminuição de custos (diminuição de rotas)

Para diminuir os custos associados ao transporte nas rotas, será útil diminuir o percurso das rotas. Com esse objetivo em consideração, podemos começar por:

- Calcular a distância dos armazéns aos clientes;
- Encontrar uma loja cliente aleatória, que não pertença a nenhuma rota, como ponto de partida e encontrar o armazém mais perto dessa loja, e começar a preencher essa rota.
- Para cada cliente calcular o próximo cliente mais próximo, que vai ser adiciono como elemento seguinte à rota;
- Repetir o processo anterior até o armazém não conseguir responder à necessidade do próximo cliente.
- Quando o armazém não for mais capaz de responder à necessidade, voltamos ao segundo passo de encontrar uma nova loja.
- Este procedimento é executado até todos os clientes estarem presentes numa rota.

É importante fazer os cálculos das rotas intercalando todos os armazéns e não fazer primeiro a rota completa para um e só depois para o próximo, porque caso contrário, poderia haver um armazém a servir um cliente que estaria muito mais próximo de um outro armazém.

Heurística de diminuição de distância entre armazém e loja

Com vista em diminuir a distância dos armazéns às lojas para poder haver um lead time mais baixo podemos distribuir as lojas da seguinte forma:

- Calcular a distância de todos os clientes a todos os armazéns.
- Em seguida, cada loja fica associada ao armazém mais próximo que ainda tenha capacidade de fornecimento.

- Quando todas as lojas estiverem associadas a um armazém fornecedor, temos encontrada a nossa solução inicial.

Meta-heurística

Apesar de termos heurísticas construtivas capazes de apresentar uma boa solução, estas podem ainda ser melhoradas, basta pensar numa ocorrência como a da figura 6, que seria problemática na aplicação da heurística de minimização de distâncias aos armazéns. Neste caso o cliente 5 estar associado á rota do armazém B ia provocar uma rota irregular o que iria aumentar bastante a distância percorrida, sendo que se este cliente estivesse na rota do armazém A, a distância percorrida ia diminuir.

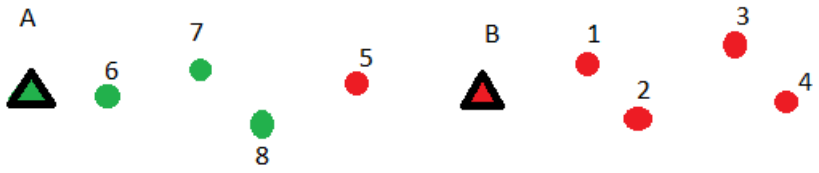

Figura 6- Exemplo de rota

Para tentar minimizar a ocorrência destas situações vamos recorrer ao uso de uma meta-heurística, o simulated annealing, um algoritmo de pesquisa local probabilístico que tem como ideia base o recozimento usado na indústria metalúrgica, que têm como objetivo encontrar o máximo ou mínimo global do espaço de resultados. No caso do nosso problema, como queremos minimizar a distância estamos em busca do mínimo local.

O algoritmo começa com uma temperatura inicial e uma taxa de arrefecimento que vai decrementar a temperatura a cada ciclo de execução. Esta temperatura é o principal fator na função de probabilidade que decide se o algoritmo aceita ou não soluções piores do que a atual.

A cada iteração o algoritmo considera um estado vizinho do estado atual e com base na qualidade, fitness, desse estado decide em avançar ou não. Caso a fitness do novo estado seja melhor, o sistema avança sempre. Se a fitness não for melhor, o sistema pode aceitar essa solução pior dependendo do valor da função de probabilidade.

Esta probabilidade é calculada da seguinte forma: $p(n) = \exp\left(\frac{1}{T} * \Delta Fn\right)$ sendo ΔFn a diferença entre a fitness da solução atual e a fitness da nova solução encontrada e T a temperatura atual que é atualizada a cada iteração segundo a seguinte equação $T = T * (1 - Cr)$ em que Cr é a taxa de arrefecimento.

A nossa implementação do algoritmo possui dois critérios de paragem:

- A temperatura ser menor do que um valor mínimo imposto.
- O número de iterações sem mudança da melhor solução ultrapassar um valor máximo imposto.

No nosso SAD temos duas aplicações do algoritmo simulated annealing, sendo a diferença entre elas a origem da solução inicial, usando cada aplicação uma heurística construtiva diferente para obter a solução inicial. A partir daí as novas soluções são geradas de duas maneiras:

- Escolher um armazém aleatório e dentro das lojas na sua rota, escolher duas e trocar o seu lugar na rota.
- Escolher dois armazéns aleatórios e dentro das lojas de cada um, escolher um cliente de cada, e se a quantidade de stock for suficiente, transferir o cliente de uma rota para o fim da outra. Soluções geradas desta maneira só são criadas de x em x iterações.

Com o algoritmo definido, e após a sua implementação em java, testamos vários parâmetros iniciais com o objetivo de encontrar a melhor solução possível, de onde obtivemos os seguintes resultados:

4.Resultados

Com a aplicação apenas da heurística construtiva de minimização da distância entre clientes e armazéns conseguimos chegar a uma solução inicial que vai ser o ponto de partida para aplicar a meta-heurística. A solução inicial tem uma distância percorrida de 1618.15 Km e segue a seguinte distribuição de lojas pelos armazéns.

Armazém	Lojas							Espaço livre
1	3	9	50	53	56	73		1
2	15	31	32	25	42	51		0
3	16							19
4	46	78						8
5	37	39	48	54	68			2
6	1	5	25	33	43	58	86	1
7	64	66	72	77	85			2
8	7	21	26	41	44	57	83	0
9	6	40	65	69	76	82		5
10	34	55	63	67	71	74		0
11	10	23	24	30				0
12	4	12	13	22	47	49	60	0
13	2	8	17	18	36	52		0
14	19	45	61	75	81	84		3
15	11	59	62	70	79			5
16	14	20	27	28	29	38	80	0

Tabela 1 – Solução inicial

Após a otimização através do algoritmo de simulated annealing conseguimos obter rotas melhores e reduzir a distância percorrida. Aqui o nível de qualidade da solução encontrada vai depender de vários parâmetros que vão influenciar o tempo de execução o que é um aspeto importante como foi falado anteriormente. Os parâmetros que podem variar são a taxa de arrefecimento, a temperatura inicial e o número de iterações que o programa faz sem encontrar nenhuma solução melhor do que a atual. Após vários testes conseguimos chegar a valores que levam a um bom desempenho, valores esses de 1,22E-10 para a taxa de arrefecimento, 3 para a temperatura inicial, e 9,9(9)E8 para o número de iterações. Com estes parâmetros conseguimos diminuir a distância percorrida de 1618,15 para 895,11 o que corresponde a uma melhoria de 44,7%. O resultado apresentado de seguida corresponde à melhor solução referida que foi encontrada em 12 minutos:

Armazém	Lojas									Espaço livre
1	9	3	75	50	73					0
2	32	42	52	58	86	1	31	15	51	0
3	16									19
4	46	78								8
5	54	11	48	68	37					2
6	33	38	25	85	43					0
7	64	13	12	44	7	5				0
8	57	41	83	21	74	71	26			1
9	65	81	19	76	6	69	40			3
10	67	39	82	63	34	55				0
11	56	2	53	18	61					1
12	47	4	77	66	49					2
13	36	35	17	8						5
14	30	10	24	23	45	84				0
15	60	79	22	59	62	70				5
16	20	80	29	72	14	28	27			0

Tabela 2 – Solução que apresenta as rotas com o melhor desempenho

Tendo em consideração o recurso “tempo de simulação” é relevante chegar a uma relação entre o respetivo tempo, que depende da taxa de arrefecimento e da temperatura inicial, e as melhorias na rota. Porque apesar de o tempo levar a uma solução melhor, existe um ponto em que este se torna elevado, podendo mesmo não se verificar uma melhoria da solução. Podemos confirmar isso no seguinte gráfico pois demostra que a relação entre o tempo e a distância segue uma função logarítmica.

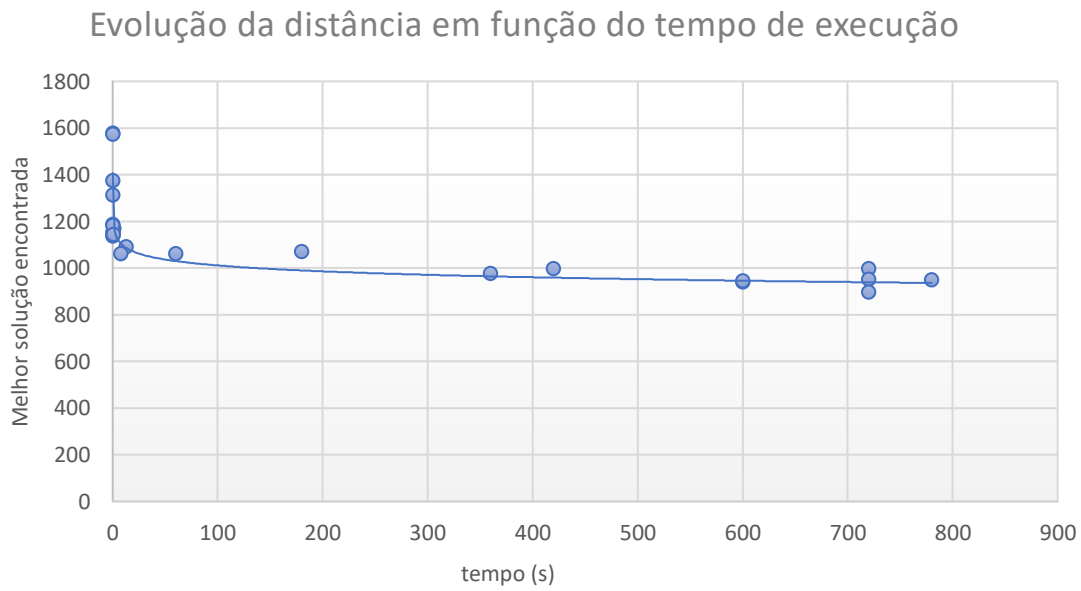

Figura 9- Relação entre o tempo de simulação e o valor encontrado para a melhor solução heurística minimizar as distâncias entre lojas e armazéns

Através de todos os testes de parâmetros que experimentamos conseguimos comprovar os efeitos que os vários parâmetros têm na meta-heurística. Com taxas de arrefecimento baixas, obtivemos soluções quase instantâneas relativamente à inicial, no entanto não apresentavam melhorias significativas. Quanto mais diminuíamos a taxa de arrefecimento, melhor ficavam as soluções, no entanto mais tempo exigiam. O parâmetro que encontramos mais eficaz para melhorar as soluções foi a taxa de arrefecimento, sendo que os melhores resultados foram encontrados quando o sistema foi iniciado com uma baixa temperatura, sendo que quando esta era mais elevada.

Nos testes utilizando a heurística que cria a solução original minimizando a distância entre lojas consecutivas de cada loja da rota os resultados foram um bocado piores. Neste método como a primeira solução é sempre gerada de maneira aleatória, os valores de distância para a primeira solução são sempre diferentes, pelo que fizemos uma média dos valores iniciais que foram apresentados por este método:

Distância inicial	média
1589,91	1484,538
1522,99	
1457,95	
1595,07	
1467,78	
1434,14	
1301,36	
1370,67	
1562,51	
1543	

A média de valores iniciais é menor do que o valor inicial da outra heurística, no entanto após a aplicação da meta-heurística estes não melhoram de maneira significativa como os valores obtidos pela outra heurística.

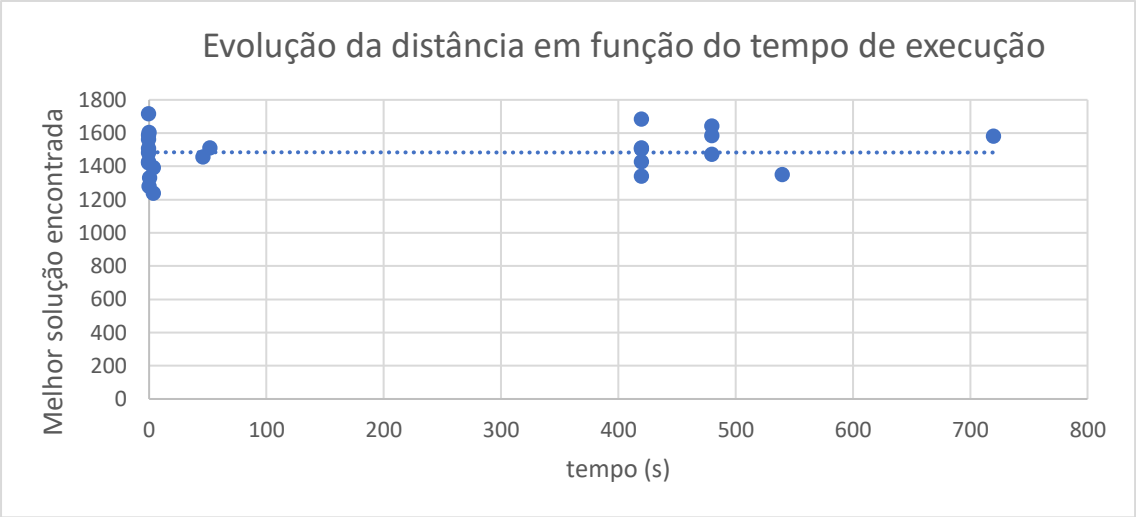

Figura 10- Relação entre o tempo de simulação e o valor encontrado para a melhor solução, heurística minimizar as distâncias entre lojas

Os resultados da aplicação deste algoritmo à heurística de minimizar as distâncias (figura 10) não se revelou muito eficaz, não apresentando melhorias significativas relativamente à média de distâncias iniciais apresentada.