Uma Aplicação de *Simulated Annealing* para o Problema de Alocação de Salas

Amanda Sávio Nascimento e Silva[1], Rudini Menezes Sampaio[2], Guilherme Bastos Alvarenga[3]

UFLA – Universidade Federal de Lavras
DCC – Departamento de Ciência da Computação
Cx Postal 37 – CEP 37200-000 Lavras (MG)
[1]amanda.nascimento@gmail.com, {[2]rudini, [3]guilherme}@dcc.ufla.br

Resumo. Este artigo apresenta uma solução algorítmica para o Problema de Alocação de Salas (PAS) utilizando a meta-heurística *Simulated Annealing* (Têmpera Simulada). A solução por meta-heurísticas foi escolhida, visto que o PAS é um problema de otimização NP-Difícil [16]. Além disso, escolheu-se a Têmpera Simulada por já haverem outros estudos com esta abordagem para o PAS. Em comparação com esses estudos, os resultados foram satisfatórios, visto que o método permite movimentos de piora como forma de escapar de ótimos locais. Para implementação do algoritmo, usou-se a linguagem de programação Java e três conhecidas instâncias do PAS para testes.
Palavras-chaves: Problema de Alocação de Salas, Simulated Annealing, meta-heurísticas, problema NP-Difícil.

An Application of Simmulated Annealing to the Class Allocation Problem

Abstract. This paper presents an algorithmic solution to the Class Allocation Problem (CAP) using the Simulated Annealing meta-heuristic. The meta-heuristic approach was chosen since CAP is a NP-Hard problem. The simulated annealing approach was chosen since there is several works of CAP with this method. We present too a comparative study with this works. The JAVA language was used to implement the algorithm and three known instances were used to test the program.
Keywords: Class Allocation Problem, Simulated Annealing, meta-heuristics, NP-Hard problem.

(Received May 10, 2005 / Accepted August 22, 2005)

1. Introdução

A área de concentração deste trabalho é o Problema de Alocação de Salas (PAS). Este problema é tratado como parte integrante do Problema de Programação de Cursos Universitários (*course timetabling*), pertencendo assim a categoria de Problemas de Programação de Horários (PPH). A conhecida dificuldade desses problemas já é uma grande motivação para este trabalho.

O PAS é um Problema de Otimização Combinatória bastante estudado. Ele é um problema NP-Difícil [8], [16], inviabilizando sua solução manual e impossibilitando sua resolução por métodos de programação matemática (métodos exatos) para grandes instâncias.

Uma vez que não é possível encontrar a solução ótima do PAS em tempo razoável para grandes instâncias, esse problema é normalmente tratado através de técnicas heurísticas e/ou algoritmos aproximativos, que apesar de não garantirem encontrar a solução ótima do problema, são capazes de retornar uma solução de qualidade em um tempo adequado para as necessidades da aplicação. Ressalta-

se que dentre as heurísticas, merecem especial atenção as chamadas meta-heurísticas, que surgiram como uma alternativa para amenizar a dificuldade que os métodos heurísticos tem de escapar dos chamados ótimos locais. Sem essa dificuldade, as meta-heurísticas podem partir em busca de regiões mais promissoras no espaço de soluções viáveis.

1.1. Objetivos e Justificativas

A meta-heurística escolhida para este trabalho foi a Têmpera Simulada (*Simulated Annealing*). A aplicação da Têmpera Simulada a Problemas de Horários tem tido relativo sucesso, conforme observado na literatura [2], [10], [22].

Para a implementação do PAS foram consideradas 3 (três) instâncias testes, sendo uma fictícia, uma baseada num caso simplificado da Universidade Federal de Lavras (UFLA) e outra baseada no caso real do Problema de Alocação de Salas do Instituto de Ciências Exatas e Biológicas (ICEB) da Universidade Federal de Ouro Preto (UFOP) [10]. Para comparação, não foi possível conseguir todas as informações relevantes a última instância, estando disponíveis apenas o número de turmas, horários das aulas e número de salas do ICEB.

Foi utilizado banco de dados para armazenamento das informações, visto que se a mudança de uma instância para outra fosse feita manualmente, a solução poderia ser insatisfatória com relação à vários aspectos, como por exemplo, salas superlotadas, aulas de disciplinas distintas alocadas em uma mesma sala em um mesmo horário, salas com restrições de uso com um número excessivo de aulas alocadas, utilização ineficiente do espaço (turmas pequenas em salas de grande capacidade), por causa do grande número de variáveis e restrições que devem ser atendidas para obtenção de uma solução ótima ou próxima da ótima.

Uma vez que o Problema de Horários (consequentemente o Problema de Alocação de Salas) é de difícil generalização, sendo desenvolvido na maioria das vezes, para atender a uma instituição específica, dado que há uma diversidade de regimes que variam de instituição para instituição, não existe na literatura um conjunto de problemas que possa ser usado na avaliação desses algoritmos. Sendo assim, pretende-se avaliar o algoritmo implementado através da observação crítica da solução gerada pelo programa, avaliando-se, por exemplo, quantas inviabilidades o programa retirou de uma solução inicial para gerar a solução final. Além disso, é feita uma observação do método proposto em relação aos métodos já existente na literatura através da comparação do desvio percentual médio em relação a melhor solução encontrada neste trabalho.

2. Problema de Alocação de Salas

O Problema de Alocação de Salas pode ser tratado como um Problema de Programação de Horários, mais precisamente como um Problema de Programação de Cursos Universitário (C*ourse Timetabling*), ou como um problema derivado deste (C*lassroom Assigment*) [4]. Para a última situação, considera-se que as aulas dos cursos universitários já têm seus horários de início e de término definidos. O problema resume-se então na alocação das aulas às salas respeitando os horários destas aulas e outras restrições.

Em [22], é usada uma metodologia que combina as meta-heurísticas Têmpera Simulada e Busca Tabu (*Tabu Search*) na resolução do PAS. Em [10], é tratado o Problema de Alocação de Salas, utilizando apenas a Têmpera Simulada.

3. Modelagem do Problema

Para a representação de uma alocação (solução) do problema é utilizada uma matriz $S = \left(s_{yw}\right)_{mxn}$ onde m representa o número de horários reservados para a realização das aulas e n o número de salas disponíveis. A turma t é alocada ao horário y e sala w, tal alocação é representada da seguinte maneira $t = s_{yw}$.

Uma sala vazia indica que a sala w está desocupada no horário y. Um exemplo simples de representação é dado pela Figura 1 [10], onde, na sala 1, os horários 1, 2 e 3 estão ocupados com aulas da turma 3, e a sala 4 encontra-se desocupada no primeiro horário.

3.1. Estrutura de Vizinhança

De acordo com [10], seja s' um vizinho de uma solução s. Para se atingir s' a partir de s, são usados dois tipos de movimentos: realocação e troca.

Horários \ Salas	1	2	3	4	5
1	3				4
2	3		1	6	4
3	3	5		6	7
4		5	2	6	7
5	12		2		
6	12	13	11	9	
7		13	11	9	10
8	**8**		11		10
9	**8**				10

Figura 1: Exemplo de Alocação de Salas

O movimento de realocar uma dada turma a uma sala que esteja vazia nos horários de realização desta aula caracteriza o Movimento de Realocação. Observa-se que tal movimento só é possível se a sala que receberá as aulas de uma turma estiver disponível nos horário de aulas da turma em questão.

Já o Movimento de Troca consiste em trocar de sala as aulas de duas turmas realizadas em um mesmo bloco de horários.

Para realização deste movimento, é essencial que nos horários envolvidos as salas estejam apenas com aulas das turmas relacionadas com a operação ou estejam vazias.

Uma solução $s' \in N(s)$, sendo $N(s)$ a estrutura de vizinhança considerada, é dita vizinha de s se ela pode ser acessada a partir desta através dos movimentos de troca ou realocação.

3.2. Função Objetivo

De acordo com [10], para avaliar uma alocação, os requisitos do problema são primeiramente divididos em duas classificações:

I. Requisitos essenciais: gerarão uma alocação inviável, caso não sejam satisfeitos. Exemplos:

A. Em uma mesma sala e horário não pode haver mais de uma aula.

B. Uma sala não pode receber uma turma cuja quantidade de alunos seja superior à sua capacidade.

C. Algumas salas têm alguns horários previamente reservados para a realização de outras atividades e nesses horários ficam indisponíveis.

II. Requisitos não Essenciais: são aqueles que, se não satisfeitos, não gerarão alocações inviáveis, contudo seu atendimento é desejável. Exemplos:

A. Certas salas têm restrições de uso e a utilização delas deve ser evitada sempre que possível.

B. Sempre que possível alocar a uma mesma sala alunos de um mesmo curso e período.

C. Utilizar o espaço das salas eficientemente, isto é, evitar alocar aulas de turmas pequenas em salas de maior capacidade.

D. Se possível, cada uma das salas deve ser deixada vazia em pelo menos um horário ao longo do dia, de forma a possibilitar sua limpeza.

Com isso, segundo [10] e [22], uma alocação (ou solução) s pode ser medida com base em duas componentes, uma de inviabilidade $g(s)$, a qual mede o não atendimento aos requisitos essenciais, e outra de qualidade $h(s)$, a qual avalia o não atendimento aos requisitos considerados não-essenciais. A função objetivo f que associa cada solução s do espaço de soluções a um número real $f(s)$ deve ser minimizada.

A Figura 2 mostra a modelagem do PAS (função objetivo e restrições).

Para que uma solução seja viável os requisitos essenciais devem ser atendidos, ou seja, $g(s) = 0$. Uma vez que nas componentes da função $f(s)$ os pesos dados às diversas medidas devem refletir a importância relativa de cada uma delas, deve-se tomar α_l muito maior que β_k, para todo l, k, de forma a viabilizar a eliminação de soluções não-viáveis.

4. Simulated Annealing

Simulated Annealing (SA), também conhecido por Têmpera ou Recozimento Simulado, foi proposto originalmente em [17], sendo um método de busca local que aceita movimentos de piora como forma de escapar de ótimos locais.

Sejam :

S = conjunto de alocações possíveis (espaço de soluções) para uma dada instância do PAS

L = número de medidas de inviabilidade de s

I_l = valor da l – ésima medida de inviabilidade

α_l = peso associado l – ésima medida de inviabilidade

K = número de medidas de qualidade de s

Q_k = valor da k – ésima medida de qualidade

β_k = peso associado k – ésima medida

Encontrar s, para:

Minimizar $f(s) = g(s) + h(s)$

Onde

$$g(s) = \sum_{l=1}^{L} \alpha_l I_l;$$

$$h(s) = \sum_{k=1}^{K} \beta_k Q_k.$$

Sujeito à:

$s \in S$;

$\alpha_l >> \beta_k, \quad \forall\, l, k$

$g(s) = 0$. de qualidade

Figura 2: Modelagem do PAS

Embora tenha sido desenvolvido há poucos anos, ela tem se mostrado uma importante ferramenta de otimização. A SA simula um método natural, fundamentado numa analogia com a termodinâmica ao simular o resfriamento de um conjunto de átomos aquecidos, operação conhecida como Recozimento (*annealing*) [17]. O termo e operação de Recozimento são amplamente utilizados na metalurgia.

O pseudocógido do algoritmo é mostrado na Figura 5. De acordo com [6], este algoritmo se decompõe em duas grandes buscas sobrepostas:

- A busca interna contém o processo de otimização. Observa-se que para uma temperatura fixa, explora-se a vizinhança aceitando ou não os movimentos que são apresentados.
- A busca externa controla o término do processo e é baseada na noção de estados resfriados. Considera-se que um estado esteja resfriado se não houver mais chances de se achar uma solução melhor ao se continuar a exploração.

O algoritmo começa a busca a partir de uma solução inicial qualquer. O laço de iterações, que caracteriza o procedimento principal, gera aleatoriamente, em cada iteração, um único vizinho s' da solução corrente s. A variação do valor da função objetivo é testada a cada geração de um vizinho. Para o teste desta variação é feito o seguinte cálculo: $\Delta = f(s') - f(s)$.

Se $\Delta < 0$, s' passa a ser a nova solução corrente. Se o vizinho gerado for pior por uma quantidade $\Delta > 0$, ele é aceito com uma probabilidade $e^{-\Delta/T}$, onde T é um parâmetro do método chamado temperatura.

Esse processo é repetido até que T seja tão pequeno que nenhum movimento possa ser aceito, ou seja, o sistema está estável. Segundo [18], a solução obtida quando o sistema se encontra nesta situação evidencia o encontro de um mínimo local.

Observa-se que a probabilidade de aceitar movimentos que degradam o valor da função objetivo decresce com a temperatura.

5. Resultados Computacionais

O programa desenvolvido para a resolução do PAS, baseado no algoritmo da Figura 5**,** foi implementado na linguagem JAVA, utilizando a máquina virtual Java j2sdk1.4.2 e o editor NetBeans, para facilitar a implementação da interface gráfica. A escolha da linguagem JAVA foi por esta ser uma linguagem multiplataforma, e por sua forte convicção na orientação a objetos, que torna a modelagem do sistema mais natural e fácil de se realizar, além de permitir a reutilização de códigos.

O sistema desenvolvido foi testado em um microcomputador PC AMD Atlhon. 1.3 MHz, com 64 MB de RAM sob o sistema operacional Windows 2000. Foram testadas três instâncias a fim de se tentar propor uma solução para o PAS. Ressalta-se que o programa foi feito de maneira flexível o que torna imediato o teste com quaisquer outras instâncias, bastando atualizar a base de dados.

Das três instâncias utilizadas, temos uma fictícia, uma baseada num caso simplificado da UFLA e outra baseada no caso real do ICEB-UFOP, como já mencionado. A tabela 1 apresenta um resumo das características das três instâncias.

Instância	Número de Salas	Número de Turmas
Instância I	6	48
Instância II	10	73
Instância III	31	254

Tabela 1: Instâncias Testadas

Para o cálculo da função objetivo das instâncias I e III foram considerados os seguintes pesos:

- Penalidade para demanda da turma maior que a capacidade da sala: 260 por aluno
- Penalidade para capacidade da sala maior que a demanda da turma: 0.01 por aluno
- Penalidade para restrições de uso das salas que foram violadas: 1.0
- Penalidade para a sala que não ficou pelo menos durante um horário do dia destinada a limpeza, sem alocar turma: 1.0

Já para o cálculo da função objetivo da Instância II, foram considerados os seguintes pesos:

- Penalidade para demanda da turma maior que a capacidade da sala: 100 por aluno
- Penalidade para capacidade da sala maior que a demanda da turma: 5 por aluno
- Penalidade para restrições de uso das que foram violadas: 1.0
- Penalidade para a sala que não ficou pelo menos durante um horário do dia destinada a limpeza, sem alocar turma: 1.0

A Tabela 2 apresenta, para cada uma das instâncias, o melhor valor encontrado para a função objetivo e o desvio percentual médio em relação à melhor solução encontrada.

Instância	Solução Final (Melhor Solução)	Desvio %
Instância I	5847.29	7.78%
Instância II	35125.00	15.6%
Instância III	27920.94	9.69%

Tabela 2: Resultados Computacionais

Para o cálculo do desvio percentual, foram realizados uma série de testes com cada uma das instâncias.

O desvio percentual em relação a melhor solução encontrada para Instância III, 9.69%, é melhor se comparado ao uso da Recozimento Simulado para o Problema de Alocação de Salas do ICEB da UFOP existente na literatura [22], cujo valor do desvio é de 10.87%. Este resultado apesar de demonstrar que o programa desenvolvido neste trabalho é eficiente e estável não permite afirmar que o programa aqui desenvolvido é melhor que o já existente, visto que como dito anteriormente, a base de dados usada nos dois programas diferem em alguns detalhes, inclusive no valor da função objetivo para a melhor solução.

A Tabela 3 mostra o número de inviabilidades relacionadas a quantidade de alunos que excede a capacidade da sala, tanto para a solução inicial como para a solução final, gerada pela SA.

Instâncias	Nº Inviabilid. Inicial	Nº Inviabilid. Final
Instância I	50	10
Instância II	445	74
Instância III	8942	16

Tabela 3: Número de Inviabilidades

Em todos os testes executados, observou-se uma redução satisfatória no número de inviabilidades do tipo demanda maior que a capacidade.

Para geração da solução inicial viável, é utilizada uma heurística, que, no entanto, não garante encontrar tal solução mesmo que existam, por ser também um problema NP-Difícil. Apesar disso, o algoritmo produziu boas alocações em um tempo mínimo.

Baseado na solução inicial, o tempo de execução do algoritmo, medido pelos testes, pode ser considerado satisfatório (20 minutos, em média).

O fato da solução final possuir inviabilidades não é de todo ruim, devido a complexidade do problema, ao grande número de restrições e a possibilidade de intervenção humana para correção manual. Isso não seria tão problemático quanto gerar toda uma alocação de boa qualidade manualmente.

Nas alocações geradas durante os testes, para a Instância III, nenhuma sala, em um mesmo horário, recebeu mais de uma aula. Em média uma sala por alocação não tinha um horário ao longo do dia destinado a limpeza, as restrições de uso das salas raramente eram violadas e aulas com horários consecutivos sempre são alocadas em uma mesma

sala. Esses fatos contribuem para a eficiência do algoritmo proposto e para a qualidade das soluções geradas.

A figura 3 apresenta a grade de horários na segunda-feira relativa à alocação inicial da Instância II, obtida pela heurística.

A figura 4 apresenta a grade de horários na segunda-feira relativa à solução final da Instância II, obtida pela Têmpera Simulada.

Ss células sombreadas indicam que na sala correspondente o número de alunos da turma alocada excedeu a capacidade da sala. Pode-se observar que estas inviabilidades são eliminadas pelo algoritmo para gerar a alocação final. As soluções geradas para as Instâncias I e III seguem o mesmo padrão dessas grades.

Para trabalhos futuros, propõe-se a inclusão de novas restrições ao modelo, aumentando a sua complexidade. Pretende-se ainda utilizar uma base de dados real e bem mais ampla, como por exemplo, o Problema de Alocação de Salas da Universidade Federal de Lavras (UFLA).

Segunda-feira													
Horários	7	8	9	10	11	12	13	14	15	16	17	18	19
Sala 489	--	--	--	--	--	6	6	--	39	39	--	53	53
Sala 481	--	--	--	--	--	--	--	--	--	--	--	--	--
Sala 482	35	--	--	--	--	--	--	--	--	--	--	--	--
Sala 483	--	--	--	--	--	7	7	68	68	70	70	--	--
Sala 484	--	--	**51**	**51**	--	--	47	47	66	66	--	--	--
Sala 485	--	--	--	--	--	--	19	19	--	--	67	--	--
Sala 486	57	57	--	--	--	14	14	--	--	--	--	--	--
Sala 487	--	--	--	--	--	--	**8**	**8**	--	--	--	--	--
Sala 488	--	--	**5**	**5**	**5**	--	--	11	11	--	--	--	--
Sala 490	--	--	--	--	--	--	62	62	--	13	--	--	--
Sala 491	--	--	9	9	10	10	--	--	--	--	--	--	--
Sala 492	--	**12**	--	--	--	--	--	--	--	--	--	--	--
Sala 493	1	1	--	--	--	--	--	--	--	--	--	--	--
Sala 494	--	2	2	2	--	--	--	--	--	--	--	--	--

Figura 3: Grade de Horários da Solução Inicial

Segunda-feira													
Horários	7	8	9	10	11	12	13	14	15	16	17	18	19
Sala 489	--	--	--	--	--	--	47	47	66	66	--	--	--
Sala 481	--	--	9	9	3	--	19	19	--	--	--	--	--
Sala 482	57	57	--	--	--	--	8	8	--	50	50	53	53
Sala 483	1	1	--	--	--	--	--	68	68	--	--	--	--
Sala 484	--	--	--	--	--	--	--	--	--	--	--	--	--
Sala 485	--	12	--	--	--	14	14	11	11	--	--	--	--
Sala 486	--	--	--	--	--	--	--	--	39	39	--	--	--
Sala 487	--	--	--	--	--	--	--	--	--	--	--	--	--
Sala 488	--	--	--	--	--	--	--	--	--	--	--	--	--
Sala 490	35	--	5	5	5	--	62	62	--	13	67	--	--
Sala 491	--	--	51	51	--	7	7	7	30	30	--	--	--
Sala 492	--	--	--	--	--	--	45	45	--	70	70	--	--
Sala 493	--	--	--	--	--	6	6	--	--	--	--	--	--
Sala 494	--	2	2	2	10	10	--	--	--	--	--	--	--

Figura 4: Grade de Horários da Solução Final

6. Referências Bibliográficas

[1] AARTS E. e KORST J., *Simulated Annealing and BoltzmannMachine*, John Wiley, 1989.

[2] ABRAMSON, D. *Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms*, *Management Science*, 37:98-113, 1991.

[3] ANDRADE, C.E. ; BATISTA, F.L.C.; TOSO, R.F. Modelo de Otimização para Transporte de Cargas em Ambientes Reduzidos, 2004. Monografia, DCC, UFLA, Lavras.

[4] BARDADYM, V. A. Computer-Aided School and University Timetabling: The New Wave, Lecture Notes in Computer Science: 1153:22-45,1996.

[5] BERNARDI, R. Aplicando a Técnica de Times Assíncronos na Otimização de Problemas de Empacotamento Unidimensional. 2001. Dissertação de Mestrado, USP, São Paulo.

[6] BIAJOLI F, L. Resolução do Problema de Programação de Jogos do Campeonato Brasileiro de Futebol. 2003. Monografia, UFOP, Ouro Preto.

[7] BURKE, E.K., COWLING, P., LANDA SILVA, J.D. and MCCOLLUM, B. *Three Methods to Automate the Space Allocation Process in UK Universities*, *Lecture Notes in Computer Science*, 2079: 254-276, 2001.

[8] CARTER, M.W. A survey if Practical Applications of Examination Timetabling algorithms. Operations Reserch, v. 34, pp. 193-202, 1986.

[9] CARTER, M.V. and TOVEY, C.A. *When Is the Classroom Assignment Problem Hard? Operations Research*, 40:S28-S39, 1992.

[10] CASTRO, O. M. Resolução do problema de alocação de salas de aula via *Simulated Annealing*. 2003. Monografia, UFOP, Ouro Preto.

[11] COSTA, D. *A tabu search algorithm for computing an operational timetable*. *European Journal of Operational Research*, 76:98-110,1994.

[12] COSTA, F.P. Programação de Horários em Escolas via GRASP e Busca Tabu. 2003. Monografia, UFOP, Ouro Preto.

[13] DOWSLAND, K.A. *Simulated Annealing*, In Reeves, C.R. (ed), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, 20-69, 1993.

[14] ERBEN, W. and KEPPLER, J. *A Genetic Algorithm Solving a Weekly Course-Timetabling Problem*, *Lecture Notes in Computer Science*, 1153:198-211, 1996.

[15] EVANS, J.R.; MINIEKA, E. *Optimization Algorithms for Network and Graphs*. USA, Marcel Dekker, USA: Marcel Dekker Inc., 1978.

[16] EVEN, S., ITAI, A. and SHAMIR, A. *On the complexity of timetabling and multicommodity flow problems*, *SIAM Journal of Computation*, 5:691-703, 1976.

[17] KIRKPATRICK, S., GELLAT, D. C., VECCHI, M. P., *Optimizations by Simulated Annealing*. *Science* v. 220, pp. 671-680, 1983.

[18] MAURI, G.R. Resolução do Problema de Programação de Tripulações de um Sistema de Transporte Público via *Simulated Annealing*. 2003. Relatório Técnico – Universidade Federal de Ouro Preto – Ouro Preto.

[19] PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combina-torialOptimization: Algorithms and Complexity*. USA: Dover Publications Inc., 1982.

[20] PARKER, R. G.; RARDIN, R. L. Discrete Optimization Computer Science and Scientific Computing Academic Press, Inc., 1988.

[21] SOUZA, M.J.F. Programação de Horários em Escolas: uma Aproximação por Metaheurísticas. 2000. Tese de Doutorado, UFRJ, Rio de Janeiro.

[22] XAVIER, A.M; ARAÚJO, C.R. Experiência com *simulated annealing* e busca tabu na resolução do problema de alocação de salas. 2001. Relatório (Apresentação PIBIC/Cnpq) – Universidade Federal de Ouro Preto, Ouro Preto.

[23] ALVARENGA, G.B.; SILVA, R.M.A.; SAMPAIO, R.M. A Hybrid Algorithm for the Vehicule Routing Problem with Time Window. INFOCOMP Journal of Computer Science, v.4, n.2, 9-16, 2005, Brazil

Procedimento Têmpera Simulada

1. Seja s_0 uma solução inicial, T_0 a temperatura inicial, *alfa* a taxa de resfriamento e $Iter_{\max}$ o número máximo de iterações para se atingir o equilíbrio.
2. $s \leftarrow s_0$; // Solução Corrente
3. $s' \leftarrow s$; // Melhor solução obtida até então
4. $T \leftarrow T_0$ // Temperatura Corrente
5. $IterT \leftarrow 0$ // Número de iterações na temperatura T
6. *Enquanto* $(T > 0)$ *faça*
7. *Enquanto* $(IterT < Iter_{\max})$ *faça*
8. $IterT \leftarrow IterT + 1$;
9. Gere um vizinho qualquer $s' \in N(s)$
10. $\Delta = f(s') - f(s)$
11. *Se* $(\Delta < 0)$ *então*
12. $s \leftarrow s'$;
13. *Se* $f(s') < f(s^*)$ *então*
14. $s^* \leftarrow s'$; //Melhor solução
15. *Fim – se*
16. *Senão*
17. Tome $x \in [0,1]$;
18. *Se* $x < e^{-\Delta/T}$ *então*
19. $s \leftarrow s'$;
20. *Fim – Senão*
21. *Fim – Enquanto*
22. $T \leftarrow alfa$ x T;
23. $IterT \leftarrow 0$;
24. Retorne s^*;

Fim Procedimento Têmpera Simulada

Figura 5: Pseudo-código da Têmpera Simulada