Cerâmica-Plan – Memorial Técnico

Arthur Henrique Sousa Cruz

Franklina M. B. Toledo

Apoio: Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)

Conteúdo

1 Introdução

O Cerâmica-Plan foi desenvolvido com o suporte da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), mediante a Bolsa de Treinamento Técnico vinculada ao projeto denominado "Empacotamento de Peças de Cerâmica em Fornos: Otimização de Espaço e Energia", número do processo 2023/09059-0. Este memorial tem o intuito de explicar detalhes técnicos do aplicativo.

O objetivo do Cerâmica-Plan é auxiliar artesãos e artesãs ceramistas na tarefa de planejar a queima das peças que produzem, ou seja, propor um arranjo para as peças de cerâmica produzidas em um forno para a queima. No contexto estudado, o forno pode ser subdividido em andares, cuja altura deve ser definida no planejamento da queima. Essas alturas são estabelecidas pela combinação dos suportes que os artesãos(ãs) têm disponíveis. O objetivo é apresentar um arranjo que melhor ocupe o espaço disponível no forno. Na Figura 1, é ilustrado um arranjo de peças no forno.

Figura 1: Exemplo de um arranjo de peças de cerâmica em um forno.

Fonte: Foto gentilmente cedida por Patrícia Degan (artesã ceramista).

A solução desenvolvida está baseada no trabalho de Belleboni (2016). O autor desenvolveu uma aplicação *web* para tratar o problema de elaboração de arranjos para queima de peças. O Cerâmica-Plan é uma aplicação independente da internet, é uma planilha elaborada utilizando a aplicação *LibreOffice Calc* (THE DOCUMENT FOUNDATION, 2023). Em termos simples, a planilha (ou aplicativo) desenvolvida incorpora elementos de interação com o usuário para gerenciar peças e fornos. Os dados fornecidos pelo usuário são então utilizados para alimentar um resolvedor (*solver*), que consiste na heurística desenvolvida por Belleboni (2016). O método de solução considera cada andar como um problema de *nesting*

bidimensional, e as peças são alocadas em cada andar utilizando a heurística *bottom-left*. A escolha desse método baseou-se em sua efetividade, conforme evidenciado na literatura (BENNELL; OLIVEIRA, 2008; DOWSLAND; VAID; DOWSLAND, 2002; BURKE et al., 2006).

O código-fonte e a licença de uso do Cerâmica-Plan se encontram hospedadas em ⟨https://github.com/thuzax/CeramicaPlan/⟩ e podem ser utilizados por artesãos sem custos financeiros, desde que respeitada a licença de uso. Na Seção 1.1, são explicados os objetivos do aplicativo desenvolvido. Em seguida, na Seção 1.2, é apresentada a estrutura do restante do texto.

1.1 Objetivos do Aplicativo

Este aplicativo tem como propósito atender aos seguintes requisitos:

1. Ser capaz de apresentar uma solução gráfica para a alocação de peças em um forno tridimensional;
2. Ser de fácil utilização;
3. Não depender de *softwares* comerciais;
4. Possuir código aberto.

Para atender aos requisitos 1 e 2, a escolha foi implementar o aplicativo como uma planilha no *LibreOffice Calc*. Ao utilizar a planilha, o aplicativo pode ser consolidado em uma única tela, garantindo assim uma experiência simples e amigável para usuários familiarizados com o uso de planilhas. O aplicativo desenvolvido se utiliza apenas de ferramentas não comerciais, atendendo ao requisito 3. Por fim, a disponibilização do código-fonte e as definições na licença de uso atendem ao requisito 4.

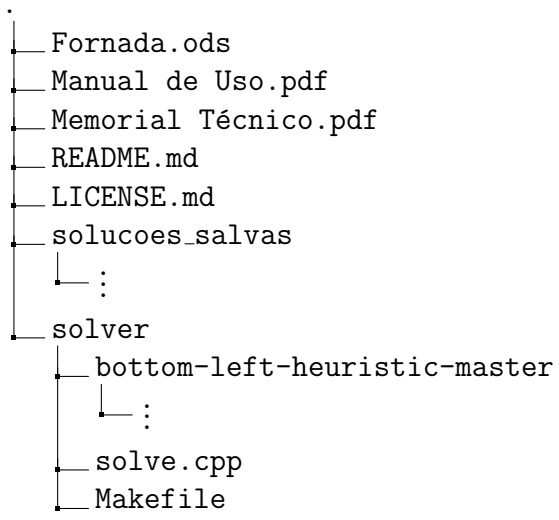
1.2 Estrutura do Texto

Na sequência deste memorial, na Seção 2, é apresentada a estrutura de arquivos do projeto, com foco na visão e utilização do usuário. Na Seção 3, é abordada a arquitetura do sistema, incluindo a comunicação entre a interface gráfica e o *solver*. Por fim, na Seção 4, é detalhado o funcionamento da interface, abordando os pseudocódigos das principais funções de interação do sistema. É importante notar que detalhes da instalação e da utilização da interface gráfica não são abordados neste memorial, visto que já foram descritos no Manual do Usuário, encontrado em: ⟨https://raw.githubusercontent.com/thuzax/CeramicaPlan/main/Manual%20de%20Uso.pdf⟩.

2 Estrutura de Arquivos e Diretórios

A estrutura do projeto, com seus diretórios e arquivos principais, está ilustrada na Figura 2. O código-fonte completo está disponível no seguinte *site*: ⟨https://github.com/thuzax/CeramicaPlan/⟩.

Figura 2: Estrutura de Arquivos e Diretórios



No diretório raiz está contido o arquivo "Fornada.ods", que é a planilha que contém o aplicativo e o código-fonte das macros a serem executadas. Também estão inclusos os documentos "Manual de Uso.pdf" e "Memorial Técnico.pdf" que são, respectivamente, o manual para uso da aplicação e este memorial. O arquivo "README.md" é uma introdução básica ao aplicativo, enquanto o arquivo "LICENSE.md" é a licença de uso e distribuição do aplicativo.

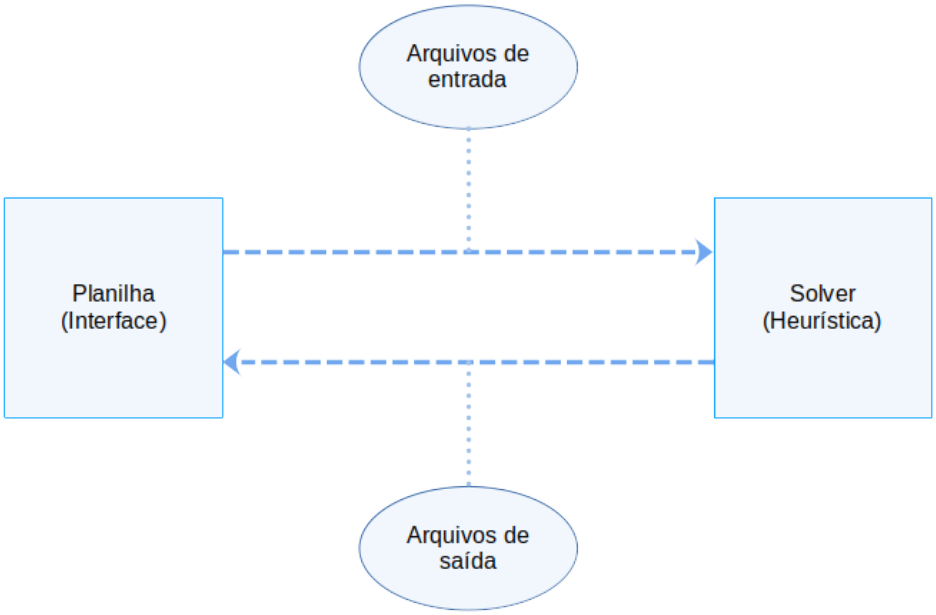
O diretório solucoes_salvas não se encontra no *site* que disponibiliza o código-fonte, pois ele é criado durante a execução da aplicação. Na primeira vez que o usuário salva uma sugestão de fornada (consulte a Seção 4.7 para mais detalhes) o diretório é criado para armazenar as soluções salvas.

Por fim, o subdiretório "solver" contém o código-fonte da heurística de solução. Dentro desse diretório encontram-se os arquivos Makefile e "solve.cpp". O primeiro é utilizado para a compilação do *solver*. O segundo faz o intermédio entre a interface e o código-fonte da heurística *bottom-left*, que está armazenada no subdiretório "bottom-left-heuristic-master". A implementação da heurística é a mesma de Belleboni (2016), com pequenas adaptações para aceitar um forno sem suportes e adicionar algumas informações na saída de dados.

3 Arquitetura e Comunicação do Sistema

A interface visa simplificar a inserção de dados, oferecendo informações sobre peças e forno. Esses dados são utilizados pelo *solver* para alocar as peças, sugerindo uma solução gráfica para a alocação das peças no forno no formato de um arquivo PDF. A interação é representada na Figura 3, onde a planilha gera e fornece arquivos de entrada ao *solver*. Este utiliza a heurística *bottom-left*, gerando um arquivo LaTeX compilado pelo *pdflatex*. Ao finalizar, a planilha abre o arquivo gerado, mostrando a solução. Arquivos temporários, incluindo os de entrada do *solver*, são excluídos após o processo.

Figura 3: Interação entre a Interface de Usuário e o Solver.



A seguir, é fornecida uma visão geral da arquitetura do aplicativo. Na Seção 3.1, são apresentados a estrutura e o processo de comunicação entre a interface e o *solver*, incluindo as entradas e saídas. Posteriormente, na Seção 3.2, são explicados os parâmetros e formato dos arquivos de entrada do *solver*. Por fim, na Seção 3.3, é explicado quais arquivos são gerados temporariamente durante a execução do *solver*.

3.1 Parâmetros do *Solver*

A chamada da heurística é feita utilizando uma linha de comando. São necessários três parâmetros de entrada: o caminho para um arquivo contendo os dados referentes às peças a serem alocadas (*pieces_file_name*), o caminho para um arquivo com informações sobre o forno (*kiln_file_name*) e o caminho do arquivo PDF que exibirá o resultado (*solution_file_name*). Na Tabela 1, são apresentados os parâmetros de entrada do *solver* e os valores que representam. Observa-se que se o arquivo *solution_file_name* existir, ele será sobrescrito, caso contrário ele será criado.

O formato da linha de comando para a chamada do *solver* é como segue:

<solver-executavel> pieces_file_name kiln_file_name solution_file_name

O campo *<solver-executavel>* representa o arquivo executável do *solver*, sendo ele gerado pelo "Makefile" com o nome "solve" em sistemas *Ubuntu*. Os parâmetros seguem o padrão definido pela Tabela 1.

Tabela 1: Parâmetros de entrada para o *solver*

Arquivo	Descrição
pieces_file_name	Arquivo contendo os dados referentes às peças. Na Seção 3.2, é detalhado o conteúdo do arquivo.
kiln_file_name	Arquivo contendo os dados referente ao forno. Na Seção 3.2, é detalhado o conteúdo do arquivo.
solution_file_name	Nome do arquivo PDF a ser gerado ao final da execução do solver.

3.2 Formato dos Arquivos de Entrada do *Solver*

Dois arquivos de entrada devem ser fornecidos ao *solver*: um arquivo contendo as informações sobre as peças (denominado aqui como *temp_pecas.txt*) e outro com os dados do forno (denominado aqui como *temp_forno.txt*). É importante observar que o nome do arquivo pode variar conforme necessário.

Tabela 2: Possíveis valores para cada peça presente em *temp_pecas.txt*

Atributo	Descrição	Valores Possíveis
Tipo	Figura que se assemelha à base da peça.	"Retângulo", "Triângulo", "Quadrado" ou "Círculo".
Descrição	Texto descrevendo a peça.	Até 100 caracteres onde espaços devem ser substituídos pelo caractere "_" (*underscore*).
Altura	Valor em centímetros representado a altura do objeto.	Inteiro maior que 0.
Largura	Valor em centímetros. Representa diâmetro, lado ou largura para, respectivamente, "Círculo", "Quadrado" ou "Triângulo", e "Retângulo"	Inteiro maior que 0.
Comprimento	Valor em centímetros. Representa o comprimento do objeto. Válido apenas para "Retângulo".	Inteiro maior que 0 se do tipo "Retângulo". Nenhum valor deve ser passado, caso contrário.
Quantidade	Quantas peças repetidas devem ser alocadas no forno.	Inteiro maior ou igual a 0.

O arquivo *temp_pecas.txt* deve ser estruturado de forma que cada linha inclua as informações de uma única peça, separadas por espaços, seguindo a ordem: *Tipo*, *Descrição*, *Altura*, *Largura*, *Comprimento* e *Quantidade*. Na Tabela 2, são apresentados os possíveis valores para cada um desses atributos.

Para utilização da heurística proposta por Belleboni (2016), cada peça deve ser categorizada de acordo com sua base. O atributo *Tipo* indica a figura geométrica da base da peça, enquanto *Descrição* oferece uma explicação breve ou o nome do objeto. As dimensões do objeto são representadas pelos atributos *Altura*, *Largura* e *Comprimento*. Conforme apresentado na Tabela 2, *Largura* pode representar diâmetro, lado ou largura, dependendo do tipo de base da peça. O atributo *Comprimento* é aplicável apenas a peças do tipo "Retângulo". Finalmente, *Quantidade* indica a quantidade de repetições da peça.

Exemplo de Arquivo 1: Descrição de uma fornada com 4 variações de peças.

```
Quadrado Peça_1 10 11 6
Retângulo Peça_2 12 15 8 3
Círculo Peça_3 6 2.5 1
Triângulo Peça_4 9 10 0
```

No Exemplo de Arquivo 1, é fornecido um caso ilustrativo com a descrição de 4 peças distintas. Nota-se que a primeira linha indica a quantidade total de peças, e nas linhas subsequentes, são apresentados os dados individuais de cada peça. Por exemplo, na segunda linha, tem-se uma peça do tipo "Quadrado" com a descrição "Peça_1", altura de 10 centímetros, largura de 11 centímetros e quantidade de repetições igual a 6.

O arquivo *temp_forno.txt* deve conter em sua primeira linha as dimensões do forno em centímetros, na seguinte ordem: Comprimento, Largura, Altura. Nas linhas seguintes, devem ser fornecidas informações referentes aos suportes que definirão a altura do andar (referidos aqui como Altura do Suporte). Os tipos de valores permitidos para Comprimento, Largura, Altura e Altura dos Suportes são apresentados pela Tabela 3. Deverá ser indicada as possíveis alturas que os suportes podem ter. A segunda linha indica a quantidade m de alturas diferentes que os suportes possuem. As m linhas seguintes devem conter o valor de cada altura em centímetros. No Exemplo de Arquivo 2, é ilustrado um exemplo de um arquivo descrevendo um forno com 5 possíveis alturas para suportes. O forno descrito contém 100 centímetros de comprimento, 120 de largura e 150 de altura. As alturas de cada andar podem ser definidas combinando suportes de 10, 15, 20, 30 e 40 centímetros.

Exemplo de Arquivo 2: Descrição de um forno com 5 suportes e 100, 120 e 150 centímetros de comprimento, altura e largura, respectivamente.

```
100 120 150
5
10
15
20
30
40
```

3.3 Arquivos Temporários

Durante a execução do *solver*, são gerados alguns arquivos temporários. O primeiro deles é uma versão adaptada da entrada, na qual os itens com quantidade igual a zero são

Tabela 3: Dados referente ao forno a ser utilizado

Atributo	Descrição	Tipo
Altura	Altura do forno em centímetros	Inteiro maior que zero
Largura	Largura do forno em centímetros	Inteiro maior que zero
Comprimento	Comprimento do forno em centímetros	Inteiro maior que zero
Altura do Suporte	Possível altura (em centímetros) para suportes que são utilizados para montar os diferentes andares. Geralmente há mais de um.	Inteiro maior que zero

removidos. Para peças com quantidade superior a um, ocorre a duplicação, uma vez que a implementação da heurística considera apenas uma peça por linha.

A primeira saída gerada pelo *solver* é um arquivo com extensão *.TEX* que contém as informações da solução. Utilizando o *pdflatex*, esse arquivo é compilado, resultando na geração de um PDF com a solução gráfica.

No Apêndice A, é apresentado um exemplo do arquivo final gerado pelo *solver*. Inicialmente, são exibidas as imagens que representam como as peças são organizadas em cada um dos andares de acordo com seu formato. Cada peça é associada a um índice específico, e após as imagens de cada andar, é apresentada a descrição das peças para cada item.

Adicionalmente, durante o processo de compilação, arquivos auxiliares e *logs* são criados. Todos os arquivos temporários gerados nesse processo, juntamente com os arquivos de entrada e outros arquivos temporários, são excluídos ao final da execução. O arquivo PDF com resultado gráfico é mantido somente se o usuário optar por salvar a fornada.

4 Funcionalidades da Interface do Usuário

Na Figura 4, é apresentada a interface do usuário aberta na aplicação *LibreOffice*. À esquerda, a partir da segunda linha entre as colunas A até F, devem ser digitados os dados referentes às peças que serão colocadas no forno. Nas colunas A e B, tem-se, respectivamente, a descrição da peça e seu tipo. Nas colunas C, D e E, são reportadas a altura, a largura e o comprimento de cada peça. Por fim, na coluna F, é definida a quantidade da peça a ser alocada no forno. Nota-se que é mantido um breve sumário dos possíveis valores para as colunas referentes às peças a partir da linha 16 da coluna H.

As informações referentes ao forno devem estar contidas entre a segunda e a quinta linhas da coluna I, sendo elas referentes aos valores da altura, largura, comprimento do forno e altura dos suportes que segurarão os andares (referidos aqui como altura dos suportes), respectivamente. Analogamente às peças, é mantido um breve sumário referente aos valores do forno a partir da linha 16 da coluna I.

Observa-se que somente os campos para a inserção de dados referentes às peças e ao forno podem ser editados pelo usuário. As outras células são protegidas e, para a edição ser feita, é necessário desativar a proteção da planilha.

A interface de usuário apresenta também dois botões: “Zerar Quantidades de Todas as

Peças" e "Gerar Fornada". Contudo, outros botões são omitidos do usuário e utilizados somente para teste e desenvolvimento: "Verificar Dados de Entrada" e "Habilita Botões". Para visualizar esses botões é necessário habilitar o modo de edição a partir do controle de formulário. A planilha com o modo de edição habilitado é exibida pela Figura 5. A implementação das funcionalidades dos botões e da planilha foram feitas como macros do *LibreOffice*, na linguagem *Basic*.

Na sequência, na Seção 4.1, é apresentada a estrutura e organização dos arquivos que contêm as macros implementadas. Na Seção 4.2, é explicado o pré e o pós-processamento executado no início e no fim de todas as macros. Em seguida, na Seção 4.3, é feita uma breve discussão sobre o botão "Habilita Botões". Posteriormente, na Seção 4.4, é detalhada a utilização das macros acionadas por eventos da planilha. Na Seção 4.5, é explicada a macro executada pelo botão "Verificar Dados de Entrada". Na sequência, na Seção 4.6, é apresentado o funcionamento do botão "Zerar Quantidades de Todas as Peças". Por fim, o funcionamento do botão "Gerar Fornada" é abordado na Seção 4.7.

Figura 4: Exemplo de planilha contendo informações relacionadas a quatro peças.

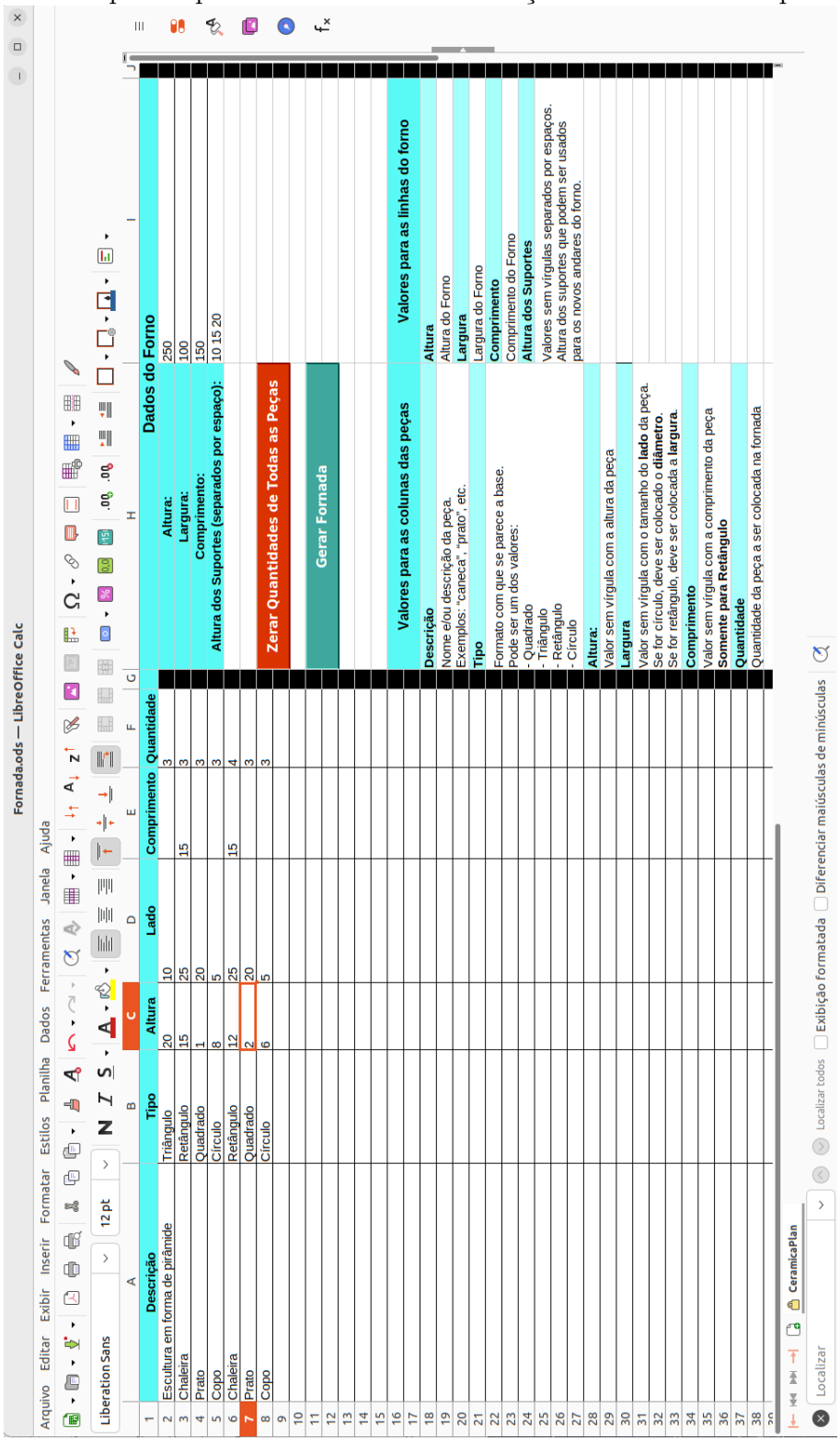
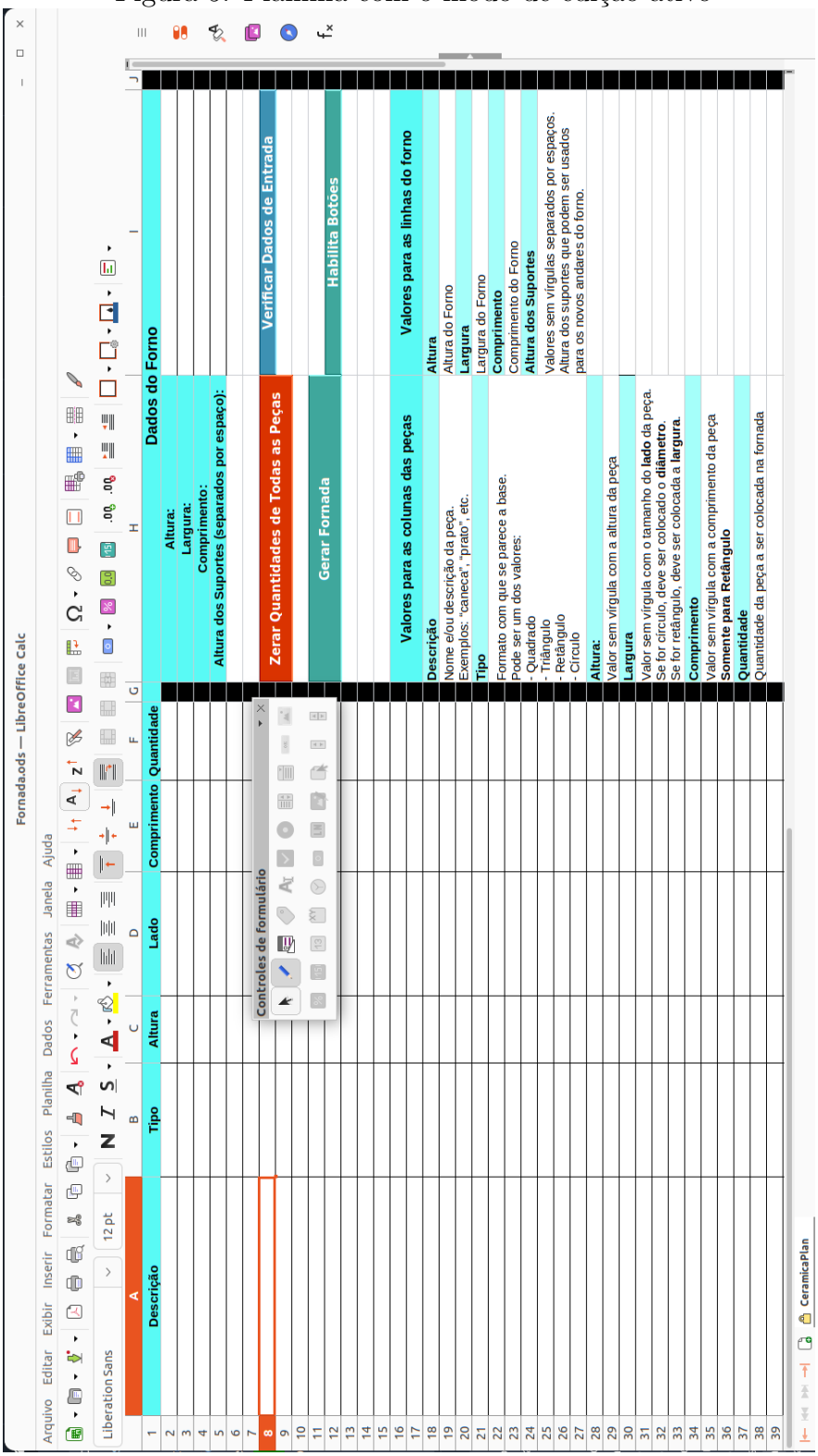


Figura 5: Planilha com o modo de edição ativo



4.1 Estrutura e Organização de Arquivos

As macros implementadas na planilha foram organizadas em doze módulos. Por conveniência, denominaremos os módulos que contêm o código-fonte das macros como arquivos. Para acessar o código é necessário utilizar o *LibreOffice*. Com a planilha que contém a aplicação aberta no *LibreOffice Calc*, selecione no menu superior à opção "Ferramentas" > "Macros" > "Editar Macros...". Isso abrirá um editor de texto. No canto superior esquerdo, haverá uma caixa de seleção. Escolha a opção "[Fornada.ods].Standard" para exibir o código-fonte da planilha. A seguir estão os nomes desses arquivos e as descrições das funções que contêm:

- $ButtonsManager$: Contém funções relacionadas aos botões, como busca e habilitação.
- $CellsManager$: Contém funções relacionadas às células, como busca, navegação e remoção.
- $ExecuteAction$: Contém funções chamadas a partir das interações do usuário com a interface, como as funções executadas ao se clicar em um botão.
- $FileManager$: Contém funções relacionadas ao gerenciamento de arquivos.
- $Globals$: Contém funções e variáveis utilizadas em diferentes contextos.
- $KilnManager$: Contém funções relacionadas ao gerenciamento dos fornos.
- $ListManager$: Contém funções relacionadas ao gerenciamento de peças na lista, como edição e verificação dos campos.
- $Messages$: Contém funções de montagem e exibição de mensagens de usuários.
- $PathManager$: Contém funções relacionadas ao sistema operacional e ao caminho de arquivos.
- $PiecesManager$: Contém funções relacionadas ao gerenciamento das peças, como a comparação entre peças e extração de valores.
- $Verifications$: Contém funções de verificação, como tipos de variáveis.
- $SolverManager$: Contém funções relacionadas ao *solver*.

4.2 Preparação e Finalização da Execução

Para evitar que duas macros sejam executadas ao mesmo tempo, e permitir alterações na planilha, ao iniciar uma macro o Algoritmo 1 é executado. Inicialmente, a proteção de célula é removida. A senha padrão para ativar ou desativar a proteção é "123456" e pode ser alterada no arquivo *Globals*. Após permitir a edição, os botões são desabilitados e a última posição da lista de peças é calculada. Por fim, como será visto nas seções seguintes, a maioria dos botões executa as verificações de invalidez e repetição, sendo necessário redefinir o vetor que armazena as marcações das peças. Observa-se que esse vetor é utilizado para

aumentar a velocidade de processamento da verificação e alterá-lo não irá modificar as células diretamente.

Ao finalizar a execução da macro, os botões precisam ser novamente habilitados e a proteção da planilha precisa ser reiniciada. Essa sequência de operações é ilustrada pelo Algoritmo 2.

Algoritmo 1: *PrepareActionExecution*

```
1 início
2 |  remove_protecao();
3 |  desabilita_botoes();
4 |  calcula_ultima_pos_lista_pecas();
5 |  reseta_marcadores_pecas();
6 fim
```

Algoritmo 2: *EndActionExecution*

```
1 início
2 |  habilita_botoes();
3 |  protege_planilha();
4 fim
```

4.3 Botão Habilita Botões

O botão "Habilita Botões" é um botão omitido do usuário e utilizado apenas para desenvolvimento. Ao ser pressionado, ele habilita todos os botões para interação. Isso é útil durante o desenvolvimento, pois alguns botões são desabilitados ao iniciar a execução de qualquer macro (consulte a Seção 4.2 para mais detalhes sobre a desabilitação ocorrida na preparação da execução).

4.4 Eventos da Planilha

Após a edição de uma ou mais células, o evento "Conteúdo Alterado" é acionado pelo *LibreOffice*. No aplicativo desenvolvido, utiliza-se desse evento para verificar a validade após a edição de uma célula. No Algoritmo 3, é apresentado o pseudocódigo utilizado para a verificação. O algoritmo recebe como entrada a célula editada. Primeiramente, a coluna *Tipo* é corrigida, permitindo pequenas variações no nome dos tipos, como falta de acentuação. Se não for possível corrigir, a célula é deixada em branco. Após isso, verifica-se se a linha da célula é inválida e, caso seja, ela é marcada utilizando vermelho.

Algoritmo 3: *AfterChange*

Entrada: célula editada C

```
1  início
2  |  L = pega_linha(C);
3  |  corrige_campo_tipo(L);
4  |  se linha_peca_invalida(L) então
5  |  |  marca_linha(L);
6  |  |  retorna;
7  |  fim
8  |  senão
9  |  |  desmarca_linha(L);
10 |  fim
11 fim
```

4.5 Botão Verificar Dados de Entrada

O botão "Verificar Dados de Entrada" é um botão omitido do usuário e utilizado apenas para desenvolvimento. Ao ser pressionado, é verificado se há algum valor inválido nos dados da lista de peças ou se há peças repetidas. Também é verificado se os dados do forno são válidos.

Os dados de uma peça são considerados inválidos se não seguirem os padrões definidos na Figura 6. Duas ou mais peças são consideradas repetições se seus valores para "Descrição", "Tipo", "Altura", "Largura" e "Comprimento" forem todos iguais.

Figura 6: Padrões de entrada para cada campo da planilha.

Valores para as colunas das peças	Valores para as linhas do forno
Descrição	**Altura**
Nome e/ou descrição da peça. Exemplos: "caneca", "prato", etc.	Altura do Forno
Tipo	**Largura**
Formato com que se parece a base. Pode ser um dos valores: - Quadrado - Triângulo - Retângulo - Círculo	Largura do Forno
Altura:	**Comprimento**
Valor sem vírgula com a altura da peça	Comprimento do Forno
Largura	**Altura dos Suportes**
Valor sem vírgula com o tamanho do **lado** da peça. Se for círculo, deve ser colocado o **diâmetro**. Se for retângulo, deve ser colocada a **largura**.	Valores sem vírgulas separados por espaços. Altura dos suportes que podem ser usados para os novos andares do forno.
Comprimento	
Valor sem vírgula com a comprimento da peça **Somente para Retângulo**	
Quantidade	
Quantidade da peça a ser colocada na fornada	

A verificação de repetições e valores inválidos é feita pelo algoritmo 4. Nas linhas 2-5,

verifica-se se a lista está vazia e, caso esteja, exibe-se uma mensagem e a execução é finalizada. Caso haja elementos na lista, nas linhas 6-11, verifica-se se há peças com valores inválidos. Se houver valores inválidos, as linhas são marcadas em vermelho; exibe-se uma mensagem de erro e a função é finalizada. Caso contrário, todas as linhas são desmarcadas (linha 11) e será verificado se há repetições (linhas 12-17). Caso haja, as linhas repetidas são marcadas em amarelo e uma mensagem indicando as repetições é exibida e a verificação é finalizada. Por outro lado, caso não haja repetições, todas as linhas são desmarcadas (linha 17). Em seguida, os dados do forno são analisados (linhas 18-23). Se houver campos inválidos, eles são marcados em vermelho, uma mensagem de erro será exibida e a execução da macro será finalizada. Caso contrário, seus campos são desmarcados na linha 23.

É importante observar que a verificação de repetições e invalidez de peças pode demandar muito tempo. Os algoritmos implementados para identificação de repetições e invalidez têm complexidades assintóticas de $O(n^2)$ e $O(n)$, respectivamente, considerando n peças. Contudo, a extração dos dados e alteração da cor das células das peças é custosa, impactando no desempenho.

Algoritmo 4: $VerifyInputValues$

```
1  início
2      se lista_vazia() então
3          mostra_mensagem_linhas_lista_vazia();
4          retorna;
5      fim
6      se tem_linhas_invalidas() então
7          marca_linhas_invalidas();
8          mostra_mensagem_linhas_invalidas();
9          retorna;
10     fim
11     desmarca_invalidas();
12     se tem_linhas_repetidas() então
13         marca_linhas_repetidas();
14         mostra_mensagem_linhas_repetidas();
15         retorna;
16     fim
17     desmarca_repeticao();
18     se forno_invalido() então
19         marca_linhas_forno_invalido();
20         mostra_mensagem_forno_invalido();
21         retorna;
22     fim
23     desmarca_forno_invalido()
24 fim
```

4.6 Botão Zerar Quantidade de Todas as Peças

Como descrito pelo nome, o botão "Zerar Quantidade de Todas as Peças" faz com que a quantidade de cada peça da lista assuma o valor 0. No Algoritmo 5, é apresentado o pseudocódigo da função que zera todas as quantidades. Primeiro é verificado se a lista está vazia (linhas 2-5) ou se há peças inválidas (linha 6-10). Em ambos os casos, é exibida uma mensagem de erro e, caso haja peças com valores inválidos, suas linhas são marcadas em vermelho. Caso haja elementos e todas as peças sejam válidas, é verificado se há repetições. Se houver, as linhas são marcadas em amarelo e uma mensagem informando das repetições é exibida (linhas 11-14). Em seguida, havendo ou não repetições, é solicitada uma confirmação do usuário antes de zerar as quantidades (linha 15). Se confirmado, os valores da coluna quantidade são zerados em todas as linhas (linhas 16-18). Caso contrário, a operação finaliza sem alterar os valores da quantidade de nenhuma das peças.

Algoritmo 5: *SetAllToZero*

```
1  início
2      se lista_vazia() então
3          mostra_mensagem_lista_vazia();
4          retorna;
5      fim
6      se lista_invalida() então
7          marca_invalidos();
8          mostra_mensagem_lista_invalida();
9          retorna;
10     fim
11     se lista_tem_repeticoes() então
12         marca_repeticoes();
13         mostra_mensagem_aviso();
14     fim
15     se confirma_acao() então
16         para i = 1, i <= tamanho_lista, i++ faça
17             zera_coluna_quantidade_na_linha(i);
18         fim
19     fim
20 fim
```

4.7 Botão Gerar de Fornada

O botão "Gerar Fornada" é responsável pela chamada do *solver* e pela sugestão de solução do problema. Inicialmente é verificado se a lista está vazia, se há peças inválidas ou se há valores inválidos para o forno similarmente à vista nas Seções 4.5 e 4.6. Nos três casos, a execução é interrompida e uma mensagem é exibida. Se houver valores inválidos para peças, as linhas relacionadas são marcadas em vermelho. O mesmo ocorre para campos do forno que contém valores inválidos.

Se não houver valores inválidos, é verificado se há repetições na lista de peças. Se houver,

elas são marcadas em amarelo e pergunta-se ao usuário se ele deseja continuar considerando apenas a primeira aparição de cada repetição. Caso não deseje, a operação é interrompida. Por outro lado, se desejar, a execução continua considerando apenas a primeira aparição de cada repetição.

Após as verificações, é feita a coleta dos dados das peças e do forno e a chamada do *solver* é realizada pelo Algoritmo 6. Para a chamada do *solver* são criados arquivos para conter os dados da peça e do forno (seguindo modelo apresentado na Seção 3.2). Além disso, o nome do arquivo de solução é gerado (linhas 2-4). Em seguida, na linha 5, o comando de execução contendo os caminhos para os arquivos de entrada e de solução é gerado e executado.

O resultado da execução é um arquivo temporário contendo a solução do problema (exemplificado pelo Apêndice A). Uma mensagem é exibida ao usuário perguntando se ele deseja salvar a solução obtida (linhas 8-9). Caso deseje, na linha 8, o arquivo PDF é copiado e renomeado de forma que seu novo nome tem o seguinte padrão: "FORNADA_<dia>-<mês>-<ano>_<hora>h<minuto>m<segundo>s". Nessa *string*, <dia>, <mês> e <ano> representam, respectivamente, o dia, o mês e o ano da data de geração da sugestão de fornada. <hora>, <minuto> e <segundo> representam, respectivamente, a hora, os minutos e os segundos do horário de geração da sugestão de fornada. O arquivo é salvo no subdiretório "soluções_salvas" no diretório raiz do projeto.

Por fim, após salvar ou não a solução, os arquivos temporários gerados contendo os dados das peças, do forno e da solução são removidos nas linhas 10, 11 e 12. Observa-se que o arquivo salvo no diretório "soluções_salva" não é removido, apenas a versão gerada pelo *solver*.

Algoritmo 6: *SolveProblem*

Entrada: conjunto de peças P, dados do forno F

```
1  início
2      pecas = cria_arquivo_pecas(P);
3      forno = cria_arquivo_forno(F);
4      solucao = cria_nome_arquivo_solucao;
5      gera_e_executa_comando(pecas, forno, solucao);
6      deseja_salvar = pergunta_se_deseja_salvar();
7      se deseja_salvar então
8          cria_copia_com_nome_padrao(solucao);
9      fim
10     apaga_arquivo(pecas);
11     apaga_arquivo(forno);
12     apaga_arquivo(solucao);
13 fim
```

Agradecimentos

Este trabalho foi desenvolvido com apoio financeiro da FAPESP (Projeto: "Empacotamento de Peças de Cerâmica em Fornos: Otimização de Espaço e Energia", número do processo 2023/09059-0). Agradecemos também Matheus Belleboni que desenvolveu o web cerâmica, sob a orientação de Franklina M. B. Toledo e Luiz Henrique Cherri. Também agradecemos a ceramista Patrícia Degan pelo apoio na definição e descrição do problema.

Referências

BELLEBONI, M. G. S. **Aplicação Web para empacotamento de peças de cerâmica em fornos tridimencionais**. 38 p. Monografia (Graduação) — Instituto de Ciências Matemáticas e de Computação – ICMC/USP, São Carlos, SP, 2016.

BENNELL, J. A.; OLIVEIRA, J. F. The geometry of nesting problems: A tutorial. **European journal of operational research**, v. 184, n. 2, p. 397–415, 2008.

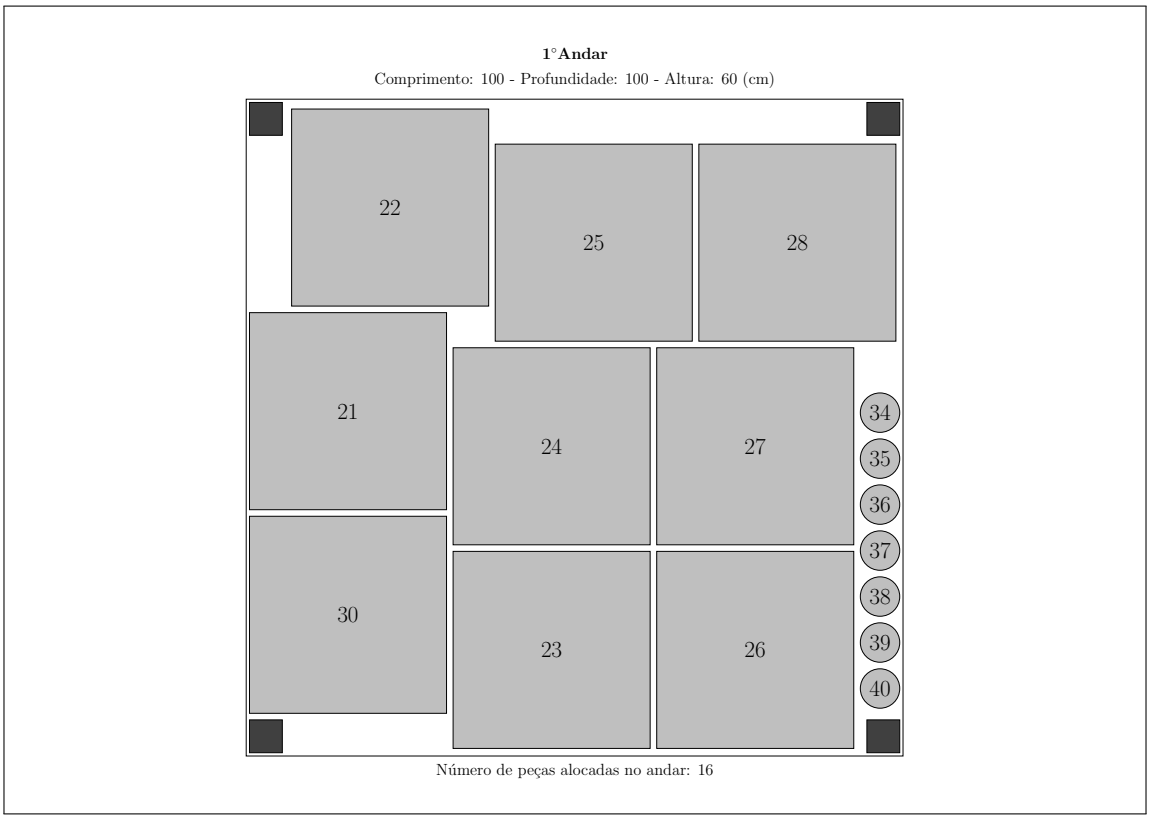
BURKE, E. et al. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. **Operations research**, v. 54, n. 3, p. 587–601, 2006.

DOWSLAND, K. A.; VAID, S.; DOWSLAND, W. B. An algorithm for polygon placement using a bottom-left strategy. **European Journal of Operational Research**, v. 141, n. 2, p. 371–381, 2002.

THE DOCUMENT FOUNDATION. **LibreOffice Calc**. 2023. Disponível em: ⟨https://pt-br.libreoffice.org/descubra/calc/⟩. Acessado em: 23/05/2024.

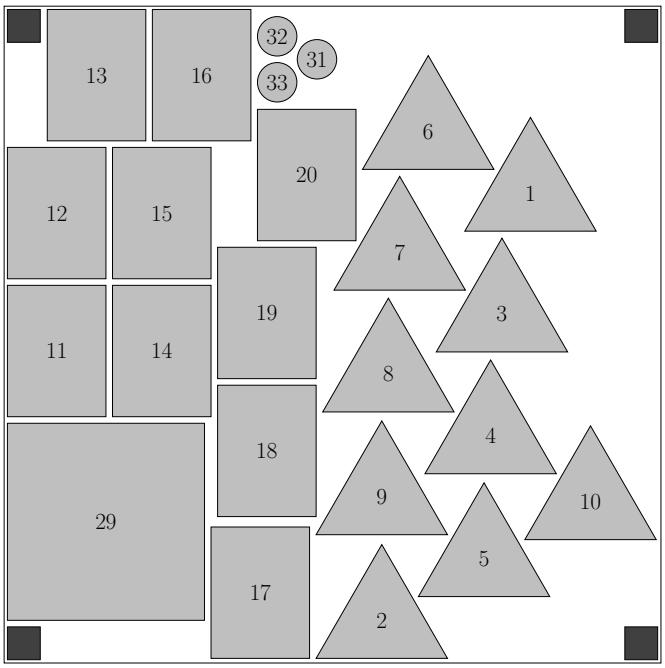
Apêndices

Apêndice A – Exemplo de Saída Gerada pelo *Solver*



2°Andar

Comprimento: 100 - Profundidade: 100 - Altura: 60 (cm)



Número de peças alocadas no andar: 24

Descrição das Peças

Número de peças alocadas: 40

Peça 1. Peça Triângulo
Peça 2. Peça Triângulo
Peça 3. Peça Triângulo
Peça 4. Peça Triângulo
Peça 5. Peça Triângulo
Peça 6. Peça Triângulo
Peça 7. Peça Triângulo
Peça 8. Peça Triângulo
Peça 9. Peça Triângulo
Peça 10. Peça Triângulo
Peça 11. Peça Retângulo
Peça 12. Peça Retângulo
Peça 13. Peça Retângulo
Peça 14. Peça Retângulo
Peça 15. Peça Retângulo
Peça 16. Peça Retângulo
Peça 17. Peça Retângulo
Peça 18. Peça Retângulo
Peça 19. Peça Retângulo
Peça 20. Peça Retângulo
Peça 21. Peça Quadrada
Peça 22. Peça Quadrada

Peça 23. Peça Quadrada

Peça 24. Peça Quadrada

Peça 25. Peça Quadrada

Peça 26. Peça Quadrada

Peça 27. Peça Quadrada

Peça 28. Peça Quadrada

Peça 29. Peça Quadrada

Peça 30. Peça Quadrada

Peça 31. Caneca

Peça 32. Caneca

Peça 33. Caneca

Peça 34. Caneca

Peça 35. Caneca

Peça 36. Caneca

Peça 37. Caneca

Peça 38. Caneca

Peça 39. Caneca

Peça 40. Caneca