PROPOSTA DE DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UM SISTEMA DE PROGRAMAÇÃO DA PRODUÇÃO PARA OS LAYOUTS DISTRIBUÍDOS: UMA ABORDAGEM COMPUTACIONAL

Proposta de desenvolvimento e implementação de um sistema de programação da produção para os layouts distribuídos: uma abordagem computacional

Shih Yung Chin
Prof. Dr. Eduardo Vila Gonçalves Filho

Departamento de Engenharia Mecânica-EESC/USP
Av. Trabalhador São-Carlense, 400, C.P.: 359, CEP: 13566-590

sychin@sc.usp.br
evila@sc.usp.br

Resumo:
Artigos referentes aos layouts distribuídos se restringem apenas na avaliação de desempenho baseado em deslocamento entre máquinas. Para lidar com o parâmetro tempo, que é um medidor de desempenho mais amplo, é preciso trabalhar com a programação da produção (PP) tornando assunto mais complexo de ser abordado, principalmente quando existem muitas máquinas e peças. Este artigo propõe um método para a elaboração e implementação de um sistema completo de PP para os layouts distribuídos. Basicamente, este é composto por um banco de dados (onde as informações sobre a condição da fábrica são armazenadas, tais como o tamanho da fila, a ociosidade das máquinas, localização do transportador, etc.) e por um conjunto de códigos de programação (forma de como o próprio sistema acessa e atualiza informações deste). Por fim, pode-se fazer real aplicação dos critérios de seleção de recursos de forma simultânea à medida que os pedidos chegam à fábrica.

Palavras-Chave: Sistema de Programação da produção, layout distribuído, critério de seleção de recursos, implementação computacional, códigos de acesso e atualização de informações do banco de dados.

Biografia: Shih Yung Chin é engenheiro de produção formado pela EESC-Universidade de São Paulo em 2002. Concluiu o curso de mestrado em 2005 nesta mesma instituição, onde realizou trabalhos sobre modelagem e simulação de Centros de Distribuição de Peças. Atualmente é doutorando no Departamento de Engenharia Mecânica na EESC, trabalhando com novas abordagens de layouts usando as ferramentas de codificação computacional e simulação.

Eduardo Vila Gonçalves Filho possui graduação em Engenharia Mecânica pela Universidade de São Paulo (1975), mestrado em Engenharia Mecânica pela Universidade de São Paulo (1982) e doutorado em Industrial Engineering - Pennsylvania State University (1988). Atualmente é professor titular da Universidade de São Paulo. Tem experiência na área de Engenharia de Produção, com ênfase em Planejamento, Projeto e Controle de Sistemas de Produção, atuando principalmente nos seguintes temas: simulação, layout, gestão do conhecimento e projeto visando manufatura e montagem

1 Introdução

Recentemente novos tipos de layouts têm surgido visando minimizar o impacto causado pela instabilidade do mercado. Os layouts distribuídos, caracterizados pela dispersão das máquinas no chão de fábrica, têm a finalidade de aproximar os diferentes tipos de máquinas entre si de modo a minimizar o percurso das peças. Isso porque quando o layout estiver funcionando ao longo do tempo, uma peça cujo lote ou o número de operações forem desconhecidos (típico de uma instabilidade), o deslocamento possa ser o menor possível.

Montreuil et al. (1993) apresentam a formulação deste tipo de layout em QAP (Quadratic Assignment Problem). Para trabalhar com este tipo de formulação, considera-se inicialmente as máquinas como sendo quadradas ou retangulares, e ao conhecer a seqüência de peças da demanda, é possível formular uma equação genérica para a obtenção do deslocamento total das peças.

Benjaafar (1995) apresenta dois métodos de solução (ótimo e heurístico) de layout distribuído, o qual também faz parte do grupo dos layouts robustos, modelando alguns parâmetros tais como a flutuação da demanda, tamanho do lote, seqüência de processamento da peça. O autor não descreve em seu artigo o layout final e muito menos como foi gerado o layout inicial. Os layouts gerados subsequentemente são obtidos também por 2-opt. Como existem diversas possibilidades de percurso entre os diferentes tipos de máquinas, o autor usou a probabilidade de ocorrência de que um determinado percurso seja o escolhido e assim obter a distância percorrida.

Aproveitando-se do modelo matemático QAP, Benjaafar e Sheikhzadeh (2000) apresentam um procedimento matemático para a geração de um arranjo modelando-o a partir das informações de instabilidade do mercado. Os resultados foram obtidos computando-se o fluxo (distância percorrida pelo lote da peça entre as máquinas). Vários outros layouts foram gerados usando a técnica pairwise exchange ou 2-opt (troca de posição duas máquinas e avalia o desempenho). O melhor layout é então armazenado pelo programa computacional. Este trabalho é bem similar ao Benjaafar (1995), diferindo apenas em alguns dados de entrada.

Recentemente alguns autores apresentaram estudos comparativos dos layouts distribuídos. Esta comparação é importante para mostrar as situações ou em que condições de operação o layout distribuído apresenta melhores resultados que o tradicional (funcional). Lahmar e Benjaafar (2005) apresentam uma série de resultados computacionais (custos de rearranjo) dos layouts (maximamente, parcial, aleatório e funcional) avaliando, por exemplo, o impacto do número de tipos de produtos no custo, a relação do nível de robustez de cada tipo de layout em relação à variação do número de réplicas de máquinas.

Neto et al. (2007) apresentam os resultados do simulador promodel avaliando a relação da variação WIP (work-in-process) de um layout distribuído e velocidade do transportador. Este trabalho não deixa de ser um desmembramento de Benjaafar (1998), porém o que diferenciam as ambas é o uso ou não da simulação. Neste trabalho, verificou-se que quanto maior a velocidade, menor é o WIP, porém a velocidade atinge um certo ponto que a redução da fila já não é verificada. No trabalho de Júnior e Filho (2007) é feito comparações entre os diversos tipos de layout distribuído em relação ao funcional trabalhando com a questão das diversas possibilidades que a peça possa ser fabricada (flexibilidade de processo) e verificou que o maximamente distribuído é o que apresenta menor fluxo de peças entre máquinas e leve vantagem em relação ao parcialmente distribuído. Neste trabalho, a obtenção da distância obtida é feita para cada seqüência de fabricação da peça e também chega a analisar a disponibilidade de máquinas.

Todos autores citados anteriormente consideram que minimizando o deslocamento, significa minimizar também o tempo de produção da peça, mas de acordo com Ham et al. (1985), o deslocamento é apenas um componente do tempo total de produção da peça. Lidar com outros componentes do tempo implica em lidar também com programação da produção, ou seja, é necessário analisar os recursos envolvidos e escolher aquele que no final resulte no menor tempo possível. Infelizmente este é um assunto ainda não abordado para este tipo de layout, pois a complexidade é muito grande em lidar com a programação, principalmente quando existem muitas peças e máquinas, o que significa que com métodos tradicionais, tais como o gráfico de Gantt, não pode ser facilmente utilizado.

Nenhum trabalho comenta como seria a implementação deste tipo de layout numa situação real de funcionamento, trabalhando sob critérios de seleção de recursos e como orientar cada

funcionário. É necessário, portanto, uma metodologia para a implementação dos critérios de seleção de forma simultânea à medida que os pedidos chegam à fábrica. Analisando cada informação que o funcionário deve inserir ao sistema PP, é possível desenvolver um modelo de armazenamento de informações (banco de dados) e entendendo a dinâmica das informações (que saem e entram neste banco de dados bem como estes são atualizados, pode-se desenvolver códigos de programação, que é, em essência, fazer buscas neste banco de dados e escolher o recurso baseado no critério de seleção. O resultado deste código aplicado ao banco de dados, enfim permite retornar dados aos funcionários sobre onde se deslocar concretizando o que foi definido pelos critérios de seleção.

2 Breve descrição sobre o funcionamento da empresa

O chão de fábrica em estudo possui duas áreas externas (como mostrado mais adiante na Figura 11). A primeira denominada de local 1 (ou L1). É neste local que a fábrica recebe os seus pedidos e é onde ocorre a conversão destes em listas aos funcionários para que a montagem de matéria-prima no transportador possa acontecer. O funcionário responsável pelos pedidos insere os dados do pedido e um sistema deve fornecer listas de tal forma que a matéria-prima a ser montada para a fabricação das peças não ultrapasse a capacidade de carga do transportador. Ao receber tais listas, o funcionário responsável pela montagem da matéria-prima (também denominado de funcionário tipo 1) sabe onde este deve atingir para o início de processamento de cada peça. Para este trabalho, todos os tipos de funcionários lidam com caixas. Nestas caixas são colocadas peças (matéria-prima, semi-acabadas ou acabadas) que por sua vez são colocadas no transportador. Cada caixa possui apenas um tipo de peça. Foi considerado que os lotes de um determinado tipo de peça caibam justamente numa única caixa. Isso quer dizer que se o transportador estiver transportando 5 caixas, quer dizer que existem lotes de 5 tipos de peças. Já no local 2 (ou L2), é uma área destinada para o depósito temporário das peças recém-fabricadas. Os funcionários responsáveis pela coleta de peças acabadas são denominados de funcionário tipo 4. Tanto o funcionário 1 quanto o funcionário 4 não existe um número limitado. Isso quer dizer que para o primeiro, assim que recebe a listagem, o funcionário 1 já parte para a entrega da matéria-prima (podendo realizar entregas mais de um funcionário 1 simultaneamente). E para o segundo, considerou-se a hipótese de que enquanto a capacidade do transportador não estiver atingida, o seguinte não deve partir de L2. Caso a capacidade do transportador tenha sido atingida, antes mesmo de retornar ao L2, o segundo parte para a coleta e assim por diante. Mais detalhes sobre a forma de como o funcionário 4 é informado para iniciar a coleta será descrita no decorrer do artigo.

Cód. Peça	**Operações/ferramental**								
1	81								
87	24	47	24	62	49	62	43	61	62
28	72	24	72	21	64	48	46	81	48
32	25	32	24	81	32	64	92	56	
38	91	75	53	54	56	74	51	13	
9	41	58	62	21	13	98	42	61	
15	13	62	31	96					
21	23	47	46	12					
4	21								
8	72	73							

Tabela 01: Exemplo de separação dos pedidos por transportador para os funcionários 1.

O funcionário responsável pela produção da peça em si é denominado de funcionário 2. Cada funcionário 2 é responsável pela sua própria máquina. Isso quer dizer que o número de funcionários 2 é o mesmo que o número de máquinas. Ele também é responsável pela montagem da matéria-prima/peça semi-acabada na máquina sob sua responsabilidade. Assim, quando inicia a operação,

traz uma caixa de peças apenas e inicia o processamento do lote daquela caixa. Assim que encerra, deixa a peça novamente na caixa.

Tendo conhecimento da existência de peças semi-acabadas e que necessitem de outras operações de fabricação, o funcionário 3 (responsável pelo transporte destas peças semi-acabadas) que está inicialmente localizado nas próprias máquinas, faz o carregamento apenas de uma única caixa (porque foi projetado para trabalhar dessa forma) e faz a condução até a máquina seguinte.

A tabela seguinte (Tabela 01) mostra a forma como o sistema deve separar o pedido por carga do transportador. De peças 1 até 38 compõem um transportador porque a carga deste foi atingido. O funcionário 1 também tem o conhecimento de que ele deve alcançar máquinas 8, 2, 7, 2 e 9 respectivamente para a entrega de matéria-prima.

Para exemplificar, foi criado um modelo de simulação em CPN tools (Redes de Petri Colorida) para verificar o funcionamento da empresa para um total de 40 tipos de peças por pedido e 20 clientes/dia (sendo um pedido por cliente), peças com até 9 operações, peso de até no máximo 10 kg, lotes de até 3 unidades de peças, transportadores até 120kg, tempos de operação em cada máquina variando de 1 a 300s, tempos de setups de ferramental de 60s, 5 transportadores 3 espalhados no chão de fábrica, 9 tipos de máquinas, total de 25 máquinas (de quantidades Qm_1=4, Qm_2=1, Qm_3=1, Qm_4=4, Qm_5=3, Qm_6=3, Qm_7=3, Qm_8=2, Qm_9=4). Não foi necessário expor os resultados do modelo, porque o que importa para este momento é conhecer quais os melhores critérios de seleção de recursos. Os melhores foram:

Para a entrega de matéria-prima:

Se a máquina estiver livre, então imediatamente é processada e se houverem mais de uma máquina, escolher a mais próxima.

Se a máquina estiver ocupada, então escolher a de menor fila.

Chamar transportador 3:

Se o transportador 3 estiver livre, então imediatamente é solicitado e se houverem mais de um transportador, escolher o transportador mais próximo.

Se nenhum estiver disponível (ou seja, todos estiverem ocupados), o primeiro que estiver livre é solicitado para o transporte.

Transporte de peças semi-acabadas pelo transportador 3 para a máquina seguinte:

Se a máquina estiver livre, então imediatamente é processada e se houverem mais de uma máquina, escolher a mais próxima.

Se a máquina estiver ocupada, então escolher a de menor fila.

Para a coleta de peças acabadas:

Escolher sempre a de maior fila que se forma após a máquina, e se houverem duas ou mais máquinas com fila de mesmo tamanho, então escolher a que estiver mais próxima do L2 ou da ultima máquina(onde ocorreu a última coleta).

3 Desenvolvimento e implementação de um sistema de PP para layouts distribuídos

3.1 Problemática

Este tópico apresenta as etapas necessárias para o desenvolvimento de um sistema de PP. A primeira etapa começa com a coleta das informações sobre o funcionamento da empresa. Dependendo do tipo de critério adotado, as informações a serem coletadas se tornam distintas. O modelo de banco de dados adotado neste trabalho para o armazenamento de informações é similar a uma lista. Em seguida, são analisadas quais devem ser os tipos de informações necessárias a serem inseridas por cada funcionário. Entendendo a dinâmica de informações que entram e saem desta lista, é possível definir códigos de busca.

Caso os critérios sejam diferentes, o procedimento de armazenamento pode sofrer pequenas alterações, mudando, por exemplo, a forma de acesso (código de programação) e a natureza das informações acessadas.

A Figura 01 apresenta um esquema geral sobre uma proposta de um sistema de PP para um layout distribuído, com I sendo o total de máquinas. As setas que saem e entram no sistema PP são informações que o sistema tanto recebe quanto envia ao sistema inteligente acoplados às máquinas, geralmente computadores.

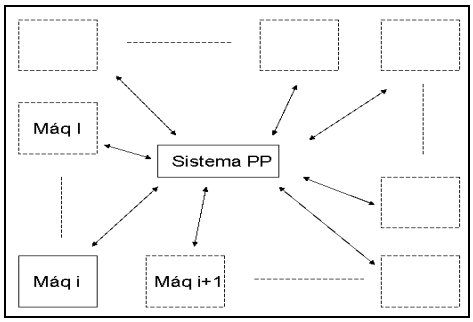

Figura 01: Esquema geral de funcionamento do sistema PP proposto.

3.2 Análise das informações inseridas por cada funcionário (proposta de inserção de informações)

São apresentados a seguir diversos passos justamente para distinguir as diferentes informações que entram e saem do sistema PP, estas que são inseridas pelos funcionários.
Passo 1:

A empresa recebe os pedidos no L1 e os funcionários (responsáveis pelo recebimento dos pedidos) devem inserir estas informações dos pedidos (podendo ser de inserção manual ou por código de barras) ao sistema PP (Figura 02).

As informações inseridas agora fazem com que o sistema PP acesse o seu banco de dados e retorne o tipo de matéria-prima, o peso da peça e as máquinas envolvidos no processamento. Com isso, o sistema PP tem informações suficientes para a criação e execução de um código de programação para separar as matérias-primas (de acordo com a solicitação do cliente) por carga do transportador. E por fim, as informações retornam aos funcionários do L1 (denominado de funcionário 1) assim que estes acessam o computador localizado em L1 (Figura 03). Lembrar que no local 1 existem dois tipos de funcionários, o funcionário 1 e outro que é responsável apenas pelos pedidos.

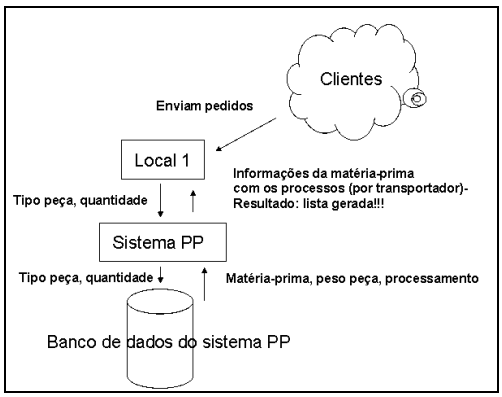

Figura 02: Fluxo de informações entre Local 1 e sistema PP.

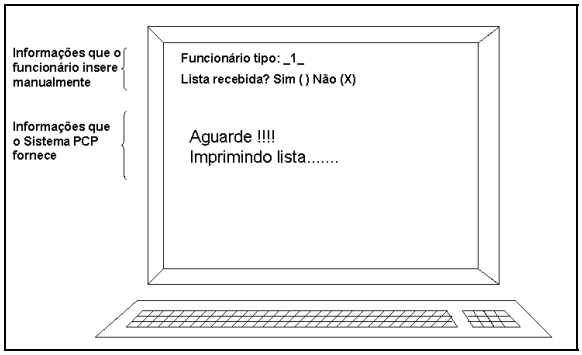

Figura 03: Visor do funcionário 1 ainda no Local 1.

O bloco de lista de peças que o funcionário 1 vai receber é similar ao da Figura 04.

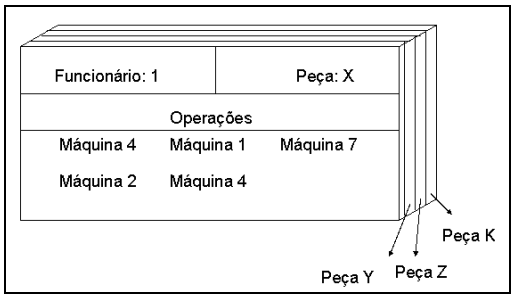

Figura 04: Exemplo de bloco de listas que o funcionário recebe.

O funcionário 2 por sua vez acessa o computador localizado na máquina sob sua responsabilidade sempre após montar matéria-prima/peças semi-acabadas. Em especial, caso for a última peça (sem mais nenhuma na fila), também ele deve acessar o computador após a execução na máquina. Ao inserir a informação do funcionário (no caso, informar que é funcionário 2), o visor do computador surge como apresentado pela Figura 05.

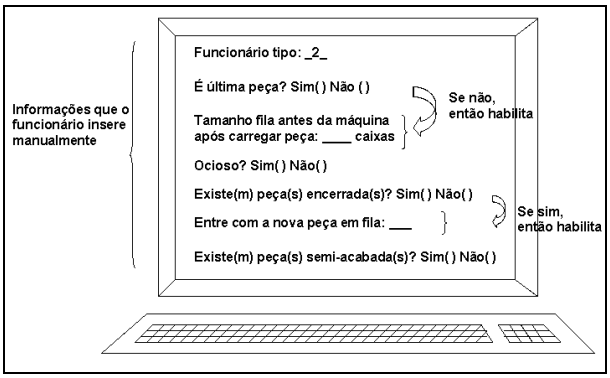

Figura 05: Visor do funcionário 2.

Quando mais de uma máquina possuem funcionários disponíveis, todos informam ao sistema preenchendo como “sim” a opção ocioso. Automaticamente o sistema PCP já vincula as máquinas com os locais (de onde a informação é enviada). Estes locais devem ser armazenados em uma listagem no banco de dados do sistema PP.

Passo 2: Até o presente momento o funcionário 1 ainda não conhece qual local ele deve atingir (já que existem muitas réplicas da mesma máquina). Assim, ao receber o bloco de listas de processamento no L1, o funcionário deve inserir a informação do tipo de máquina da primeira operação da primeira peça da lista no computador (localizado no L1), no caso, é máquina 4. No visor do computador deve conter as seguintes informações (ver Figura 06):

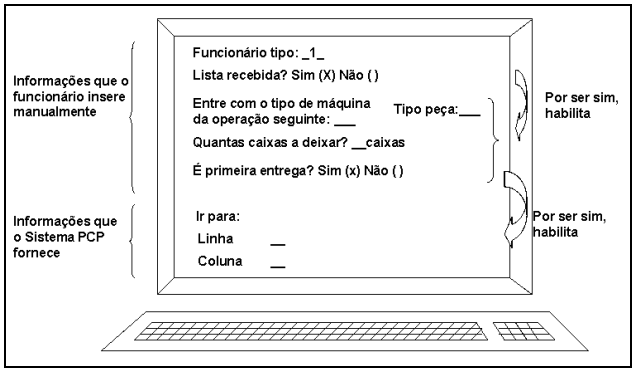

Figura 06: Visor do funcionário 1 no local 1 ou nas máquinas.

Assim que o funcionário insere as informações no visor, o sistema PP faz uma busca no seu banco de dados e verifica qual máquina 4 está disponível e que seja a mais próxima do L1. O sistema PP faz uma busca e escolheu a que está localizada na linha 7 e coluna 3, por exemplo, então o visor fornece “ir para linha: 7 e coluna: 3”. No computador da máquina 4, o funcionário 1 acessa o sistema

e informa a peça seguinte e a sua primeira operação, porque este mesmo funcionário ainda precisa continuar fazendo entrega de matéria-prima, no caso a segunda peça é a peça 23, por exemplo. Considere que a primeira operação dela é 8, e portanto com esta informação, o sistema PP faz uma busca em seu banco de dados e retorna o quanto ele deve se deslocar a partir da posição onde ele se encontra. O visor do computador da máquina 4, portanto, fica (ver Figura 07):

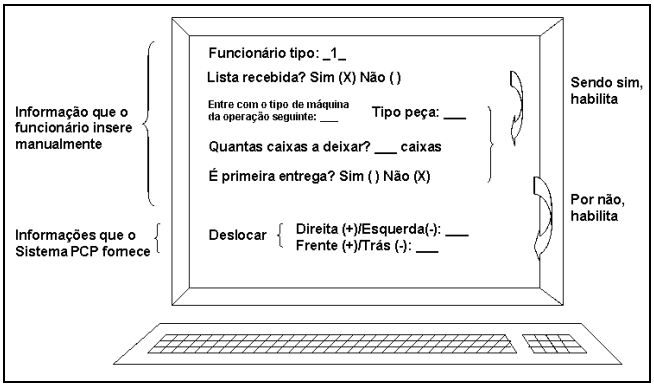

Figura 07: Visor do funcionário 1 na máquina 4.

Passo 3: Toda vez que o funcionário 2 encerra alguma operação (ou acabou de encerrar a última peça), é necessário informar ao sistema, atualizando assim a fila que se forma antes da máquina, a que se forma depois dela e sobre a disponibilidade do funcionário 2 em si. Além disso, ele informa ao sistema que uma peça semi-acabada está disponível na máquina e necessita de um transportador para a locomoção até a máquina seguinte, como mostrado na Figura 08. É importante lembrar que ao inserir funcionário tipo 1 ou 2, as informações que aparecem no visor são distintas.

Ao inserir a informação de condição “não encerrada”, a informação é enviada ao sistema PP e armazenada. Neste mesmo instante, pode ser que mais de um funcionário tenha informado sobre a condição de outras peças. Todas as informações são enviadas ao banco de dados e vão sendo atualizadas.

Passo 4: Simultaneamente ao funcionário 1, existirão peças sendo processadas pelo funcionário 2 e que necessite de transporte até as máquinas seguintes. No momento em que algum transportador intermediário (responsável pelo transporte entre as máquinas, também denominado de funcionário 3) estiver disponível, ele acessa o computador localizado nas próprias máquinas e por ter função diferente dos outros, o sistema PP já o reconhece e fornece o visor diferente (é importante lembrar que inicialmente os transportadores intermediários já se encontram nas máquinas).

Ao informar como sendo 3 e informar como “não carregado”, o banco de dados então faz uma série de comparações, baseada na menor distância das máquinas (caso houverem mais de uma máquina com peças semi-acabadas) em relação à máquina onde o funcionário 3 se encontra, e o menor é retornado ao visor do computador. Pode acontecer de durante dois ou mais acessos ao computador não tenham peças semi-acabadas, fazendo com que aparentemente trave o sistema, isso porque o funcionário 3 não tem referência sobre a freqüência (momento) de acesso. Esse problema tende a diminuir quanto menor for o número de transportadores, mas mesmo assim recomenda-se constantes acessos. Agora que está carregado, acessa novamente o computador da máquina onde ocorreu o carregamento, e ao informar a operação seguinte, o sistema faz a análise do banco de dados e informa que tipo de deslocamento que ele deve realizar.

Portanto o visor do funcionário 3 fica (ver Figura 08):

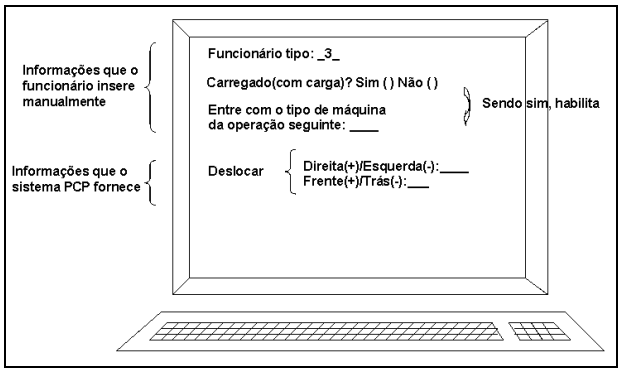

Figura 08: Visor do funcionário 3 após carregar peça semi-acabada.

Após o descarregamento da peça semi-acabada, acesssa novamente o computador e ao informar como "não carregado", o sistema PP já sabe que o funcionário 3 está disponível e em que local ele se encontra fechando assim o ciclo.

Passo 5: E por fim, quando a peça estiver em sua última operação (já concretizada), o funcionário 2 informa ao sistema PP que ele é funcionário 2 e que a peça está encerrada (ver novamente a Figura 05). Como os funcionários 2 processam e 3 transportam um tipo de peça por vez, é importante apenas informar a nova peça em fila, isso porque o banco de dados já possui registros das peças anteriores em fila de peças acabadas que se forma após a máquina

Passo 6: Como há sempre funcionários disponíveis no local 2 (funcionário tipo 4), este acessa computadores neste local e informa o tipo de funcionário que ele é. O primeiro funcionário que sai de local 2 sempre checa a existência de peças acabadas pelo sistema, já os demais são chamados pelo sinal verde emitido ao local 2. Isso porque do segundo funcionário 4 em diante não sabe quando acessar o sistema para iniciar a coleta de peças (não se tem referências). Ao inserir apenas como funcionário tipo 4, o sistema PP deve fazer busca automática e fornecer apenas o local de destino para onde ele deve atingir. Além disso, o sistema até conhece a carga do transportador porque conhecendo tamanho de fila de peças encerradas, o peso já é buscado e calculado automaticamente para o lote em fila. O visor é similar ao da Figura 09.

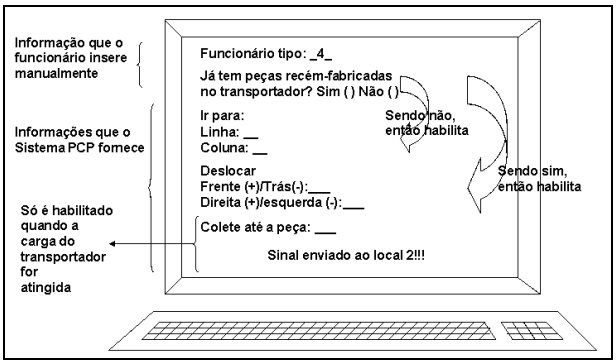

Figura 09: Visor do funcionário 4 em qualquer computador.

3.3 Proposta de um modelo de armazenamento de informações (banco de dados)

Os tópicos anteriores simularam as entradas de informações de todos os funcionários envolvidos na produção das peças. São informações que são constantemente atualizadas à medida que os pedidos dos clientes chegam ao local 1. É possível agora projetar um método de armazenamento e manipulação de informações do banco de dados do sistema PP, que é uma outra proposta deste trabalho. Em essência, o proposto neste trabalho é uma lista (tabela) composta por uma série de informações, algumas fixas (máquina tipo e local) e outras variando ao longo do tempo (demais).

Toda essa dinâmica no que diz respeito ao acesso e atualização das informações do banco de dados do sistema PP pode ser representada por códigos de programação, ou seja, o sistema PP, por meio dos códigos de programação, acessa/atualiza as informações do seu banco de dados. Esta

dinâmica é composta na realidade por 4 códigos, sendo um código para cada tipo de funcionário. Isso quer dizer que é possível no decorrer do tempo que vários códigos acessem a mesma lista simultaneamente, ver sub-tópico 3.4.

Para iniciar a proposta de um modelo de banco de dados, inicia-se com um exemplo das informações da empresa (a Tabela 02 ilustra um exemplo de lista após um período t de funcionamento onde são ilustradas as variáveis dos critérios adotados), e representa justamente o arranjo físico da Figura 11.

Alguns valores da Tabela 02 são importantes de serem citados. Pode acontecer que não exista fila que se forma antes da máquina, porém a situação da máquina estar é ocupada. No caso da tabela, coincidiu de não ter filas e a máquina estar livre, mas nem sempre isso acontece. Já o inverso, é verdadeiro, ou seja, quando estiver livre nunca terá fila.

Já outro valor importante é que quando se tem fila de peças acabadas, sempre tem algum tamanho de fila, e por isso, deve-se informar os tipos de peças que se acumulam após a máquina. A representação da Tabela 02 adotada neste trabalho e é justamente a forma como os códigos trabalham é similar a Figura 10:

1 1 3 1 10 1 0 2 0 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 0 5 0 0 0 0 0 0 0 0 0 0 0
2 1 1 0 0 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 10 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 3 4 32 46 0 0 1 2 1 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 0 10 0 1 0 4 5 76 36 29 0 2 2 1 1 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0

Figura 10: Adaptação da Tabela 02 para ser lida pelos códigos.

Ao receber o(s) pedido(s), o funcionário 1 verifica a primeira operação da primeira peça (máq. 5, por exemplo). Com esta informação, o sistema deve percorrer a primeira coluna (máq. tipo) e buscar todas as réplicas referentes a esta primeira operação. Ao encontrar a réplica, deve também fazer uma busca na coluna de “livre/ocupada” e após as buscas, conclui que existe uma máquina livre (localização x=2 e y=3).

Lembrar que a informação sobre a condição livre/ocupada foi previamente inserida pelo funcionário tipo 2. O que era nulo agora passa a ser 1 (ocupada) e a distância percorrida é obtida a partir de : De→ Para : (Xdestino-Xorigem) → (Yorigem-Ydestino).

Portanto a distância percorrida é (|2-4|+|2-3|=3). Como não haviam peças sendo processadas na máquina 5 no momento em que o funcionário chegou até ela, o número de caixas passa a ser o que levou menos 1. Se o funcionário levar apenas 1 caixa (como foi considerado neste trabalho), a fila antes da máquina 5 permanece nula, já que ela mesma já será processada na máquina restando nenhuma caixa em fila.

Ao analisar a segunda peça para a realização da primeira operação (máq. 1, por exemplo), verifica-se que existem duas máquinas 1 e ambas ocupadas. Como o critério adotado foi o de menor fila que se forma antes da máquina 1, a coluna a buscar agora é “tam. fila antes”. Nota-se que o menor é 2, e portanto o deslocamento passa a ser (|1-2|+|3-3|=1). O tamanho da fila antes da máq. 1 (local 13) agora passa a ser atualizado para dois mais as caixas que o funcionário 1 levou (considerando que durante este período o funcionário 2 não tenha terminado de processar a peça na máquina.

Note pelo código do apêndice 1 (que é outra parte da proposta deste trabalho, referente a forma como as informações são lidas e atualizadas) que o banco de dados do sistema PP é acessado pelos códigos só após quando o funcionário 1 inserir primeiro as informações de cada entrega para que o sistema PP assim faça a busca no banco de dados atualizados. Caso contrário, corre sério risco de ler dados desatualizados, já que muitos funcionários acessam e atualizam a mesma tabela durante este período de inserção de informações. É importante lembrar que ele acessa o computador em local 1(após receber o bloco de listas) e os demais acessos ocorrem só depois que deixar a matéria-prima na máquina seguinte. Os códigos foram implementados em linguagem Pascal 7.0, o que indica que não é necessário um programa super avançado.

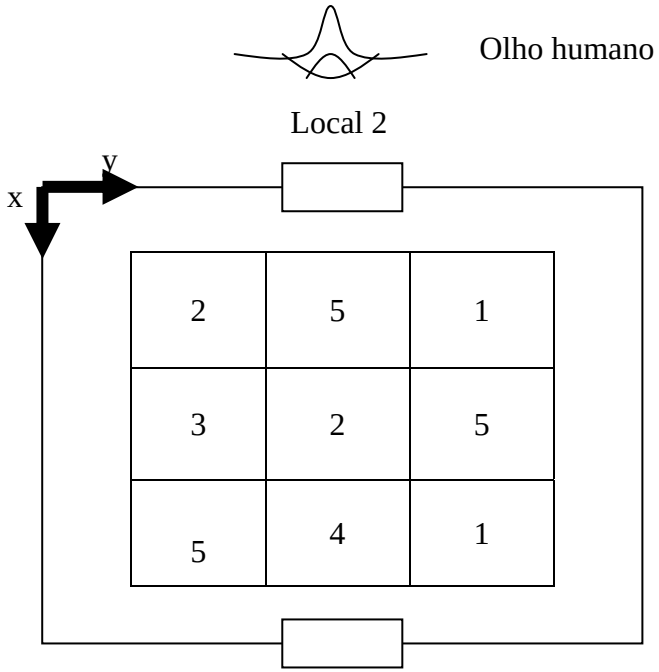

Figura 11: Exemplo de um layout distribuído.

Máquina Tipo	Local (xy)	Livre (0) / ocupada(1)	Localização do transportador 3 (10=ausente/ 1=carregado/ 0=descarregado)	Peças semi-acabadas depois da máquina (0=não tem; 1=tem)	Peças Acabadas (0=não tem; 1=tem)	Tamanho fila antes (em caixas)	Tamanho fila depois de peças acabadas (em caixas)	Acúmulo de tipos de peças acabadas [mais antigo→mais recente]	Lote de cada tipo de peça acabada[mais antigo→mais recente]
1	13	1	10	1	0	2	0	[0]	[0]
1	33	1	1	1	0	5	0	[0]	[0]
2	11	0	0	0	1	0	3	[7, 2, 7]	[2,1,3]
2	22	1	10	1	1	2	1	[97]	[2]
3	21	0	10	0	0	0	0	[0]	[0]
4	32	1	10	1	1	8	3	[4, 32, 46]	[1,2,1]
5	12	1	10	0	0	6	0	[0]	[0]
5	23	0	10	0	1	0	4	[5, 76, 36, 29]	[2,2,1,1]
5	31	1	1	0	0	5	0	[0]	[0]

Tabela 02: Informações coletadas no chão de fábrica após t período.

3.4 Exemplo de aplicação dos códigos de programação

Considere que o funcionário 1 deve entregar a matéria-prima para a fabricação de peças. Este, portanto, deve visualizar apenas as primeiras operações, começando com 5→ 1. Ao levar a peça até a máquina em x=2 e y=3, esta passa a estar na condição de ocupada e o tamanho de fila que se forma antes da máquina 5 ainda continua sendo nula porque a caixa que foi levada à ela já está sendo "usada". A Figura 12 mostra como os dados são "inseridos" pelo funcionário 1 e as Figuras 13 e 15 mostram como as informações são atualizadas no banco de dados. A Figura 14 ilustra um outro exemplo, que é a continuação da entrega da matéria-prima às máquinas, e assim por diante.

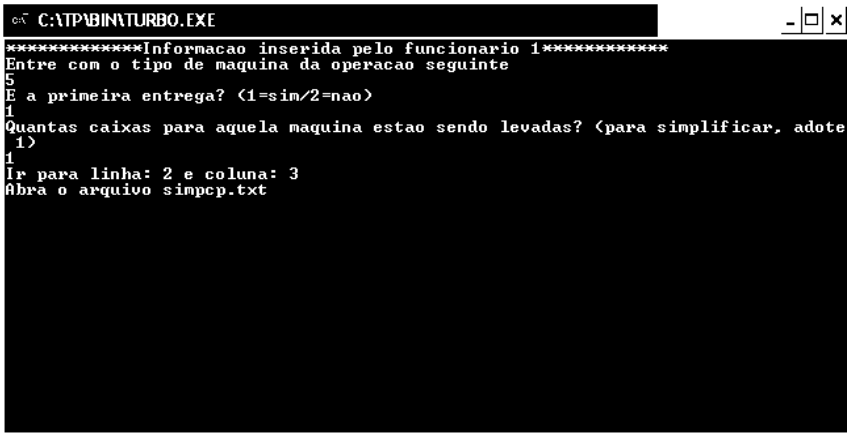

Figura 12: Exemplo de como os dados são inseridos pelo funcionário 1.

1 1 3 1 10 1 0 2 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 0 5 0 0 0 0 0 0 0 0 0 0
2 1 1 0 0 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 10 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 3 4 32 46 0 0 1 2 1 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 1 10 0 1 0 4 5 76 36 29 0 2 2 1 1 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0

Figura 13: Ilustração das atualizações do banco de dados.

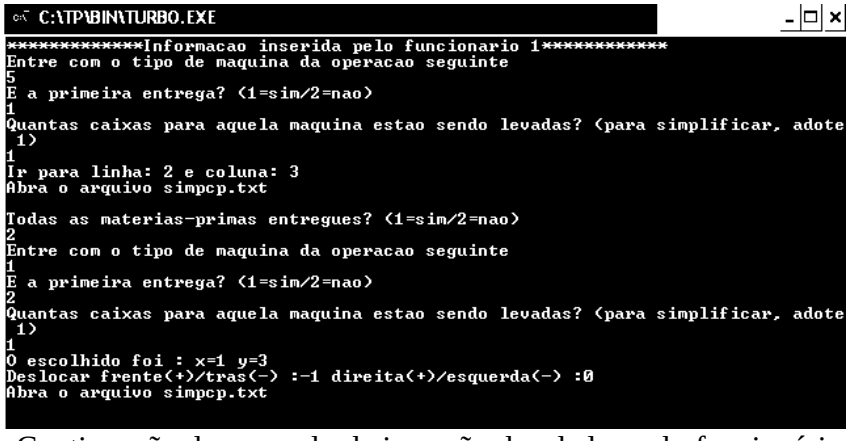

Figura 14: Continuação do exemplo de inserção dos dados pelo funcionário 1.

1 1 3 1 10 1 0 3 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 0 5 0 0 0 0 0 0 0 0 0 0
2 1 1 0 0 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 10 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 3 4 32 46 0 0 1 2 1 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 1 10 0 1 0 4 5 76 36 29 0 2 2 1 1 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0

Figura 15: Ilustra atualização do banco de dados.

O código para o funcionário 2 está apresentado no apêndice 2. Considere que o funcionário 2 da máquina 1 localizado em x=3 e y=3 tenha acessado o computador após o carregamento da caixa, ver Figura 16. Para simplificar, o banco de dados iniciado seja igual ao da Figura 10. O número de caixas em fila passa a ser reduzido a um (o banco de dados imediatamente atualiza o novo valor do número de caixas em fila (agora passa a ser 4 caixas), ver Figura 17.

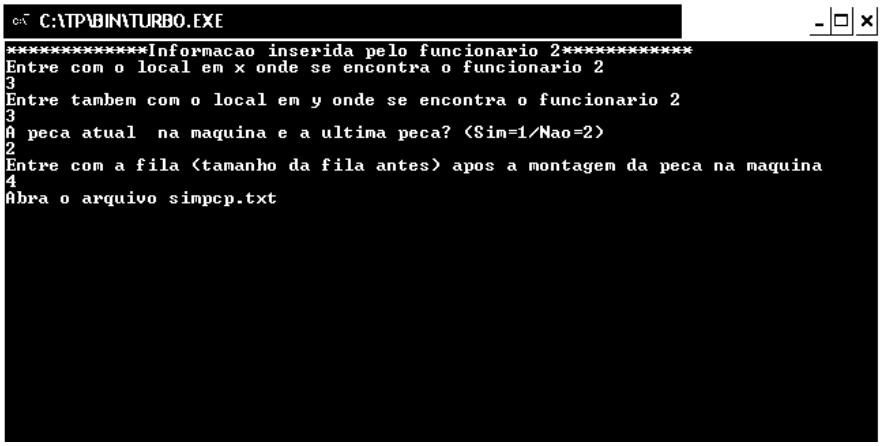

Figura 16: Exemplo de inserção de informações do funcionário 2.

1 1 3 1 10 1 0 2 0 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 0 4 0 0 0 0 0 0 0 0 0 0 0
2 1 1 0 0 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 10 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 3 4 32 46 0 0 1 2 1 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 0 10 0 1 0 4 5 76 36 29 0 2 2 1 1 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0

Figura 17: Ilustra atualização do banco de dados.

Foi necessário já efetuar atualização porque durante este período, outros funcionários necessitam atualizar também o mesmo banco de dados. Continuando a inserção das informações pelo funcionário 2, segue como mostrada na Figura 18 e sua respectiva atualização na Figura 19.

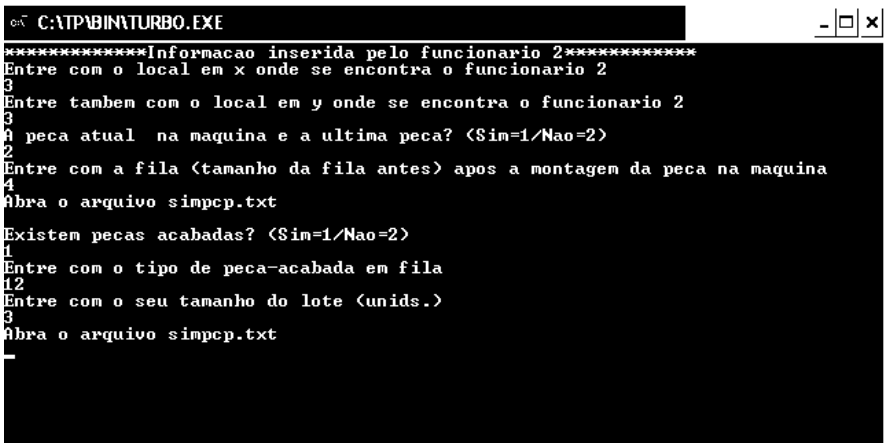

Figura 18: Continuação da inserção das informações pelo funcionário 2.

1 1 3 1 10 1 0 2 0 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 1 4 1 12 0 0 0 0 3 0 0 0 0
2 1 1 0 0 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 10 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 3 4 32 46 0 0 1 2 1 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 0 10 0 1 0 4 5 76 36 29 0 2 2 1 1 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0

Figura 19:Ilustra atualização do banco de dados.

Existem duas possibilidades quando o funcionário 2 encerra a sua operação, como dito anteriormente. A primeira possibilidade é que enquanto um funcionário tipo 1 faz a entrega de matéria-prima, um funcionário tipo 2 (localizado no local 22) acabou de realizar o processamento da peça porém esta ainda necessita de operação em máquina seguinte. Note que não é necessária uma coluna que atualize o tamanho de fila de peças semi-acabadas porque a decisão que o transportador toma não é baseada neste tipo de fila, mas sim na distância. Porém é necessário informar à coluna "peças acabadas" sobre a existência de peças necessitando de operações seguintes. Um código deve checar nesta coluna a existência de "1". Agora, o passo seguinte é necessário checar a coluna de localização do transportador e que esteja disponível. Verifica-se que o único transportador disponível está no local 11. Portanto o deslocamento feito pelo transportador até 22 é (|1-2|+|1-2|) igual a 2. Esta coluna é atualizada {em vez do local 11 ser 0 (transportador descarregado), agora passa a ser 10 (sem transportador) e o local 22 de 10 passa a ser 1 e depois 0 (após descarregamento). Com o descarregamento, a coluna "tam. fila antes" deve ser atualizada fazendo acrescentar o tamanho de fila no local 22.

Para testar o exemplo, o código referente ao funcionário 3 está apresentado no apêndice 3 e a simulação das informações inseridas pelos funcionários 3 bem como as suas respectivas atualizações do banco de dados estão apresentadas nas Figuras 20 à 26.

Figura 20: Inserção das informações pelo funcionário 3.

```
1 1 3 1 10 1 0 2 0 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 0 5 0 0 0 0 0 0 0 0 0 0 0
2 1 1 0 0 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 10 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 3 4 32 46 0 0 1 2 1 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 0 10 0 1 0 4 5 76 36 29 0 2 2 1 1 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0
```

Figura 21: Ilustra atualização do banco de dados.

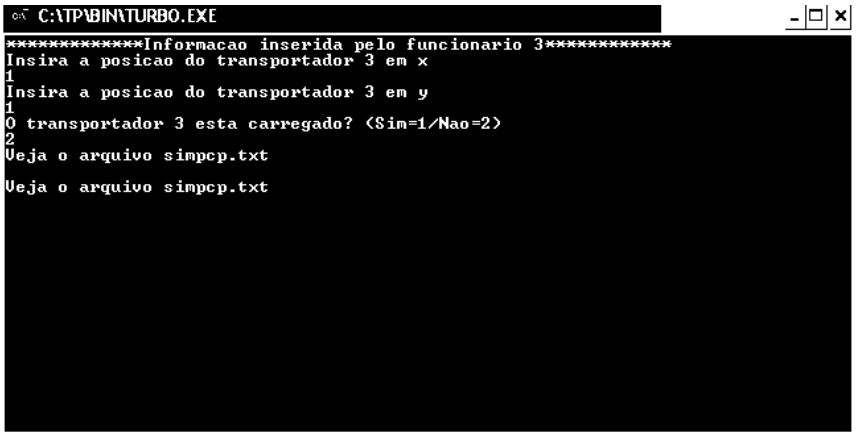

Figura 22: Inserção das informações pelo funcionário 3.

1 1 3 1 10 1 0 2 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 0 5 0 0 0 0 0 0 0 0 0 0
2 1 1 0 10 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 10 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 3 4 32 46 0 0 1 2 1 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 0 10 0 1 0 4 5 76 36 29 0 2 2 1 1 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0

Figura 23: Ilustra atualização do banco de dados.

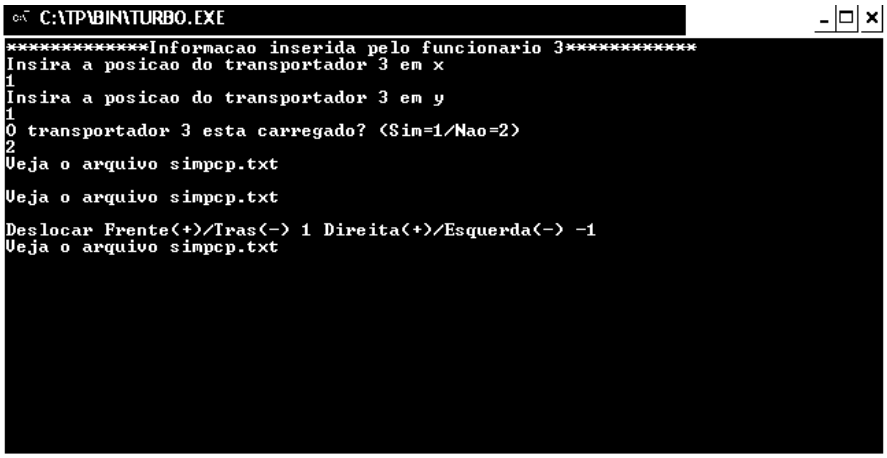

Figura 24: Inserção das informações pelo funcionário 3.

1 1 3 1 10 1 0 2 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 0 5 0 0 0 0 0 0 0 0 0 0
2 1 1 0 10 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 1 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 3 4 32 46 0 0 1 2 1 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 0 10 0 1 0 4 5 76 36 29 0 2 2 1 1 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0

Figura 25: Ilustra atualização do banco de dados.

Ao informar sobre onde este deve chegar (carregado), por exemplo, máquina 5, resulta em (ver Figura 26):

1 1 3 1 10 1 0 2 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 0 5 0 0 0 0 0 0 0 0 0 0
2 1 1 0 10 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 10 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 3 4 32 46 0 0 1 2 1 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 1 1 0 1 0 4 5 76 36 29 0 2 2 1 1 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0

Figura 26: Ilustra atualização do banco de dados.

Assim, ao descarregar, o funcionário 3 acessa o computador e informando sobre a condição de descarregado, automaticamente o sistema sabe que este está livre, e assim fechando o ciclo.

Uma segunda possibilidade é quando a peça processada é justamente a peça recém fabricada e não necessita mais de nenhuma operação. Neste caso, o código acessado é do apêndice 4. Este deve checar a coluna “peças acabadas” e escolher os que tiverem como valor 1. Depois disso, analisar a coluna “tam. fila depois de peças acabadas” (porque é baseada no tamanho desta fila que o sistema escolhe e informa ao transportador 4 onde deve-se realizar a coleta de peças) e escolher o maior deles. Caso houverem dois de mesmo tamanho, escolher a mais próxima. Ver exemplos em Figuras 27 à 30. Ao coletar a primeira caixa, se ainda não atingiu a carga do transportador e ainda tiver nesta máquina peças acabadas, continua a coleta sem a necessidade de fazer busca pelo banco de dados sobre qual máquina possui peças acabadas.

Para ilustração, foi adotado que a carga do transportador foi de 120kg.

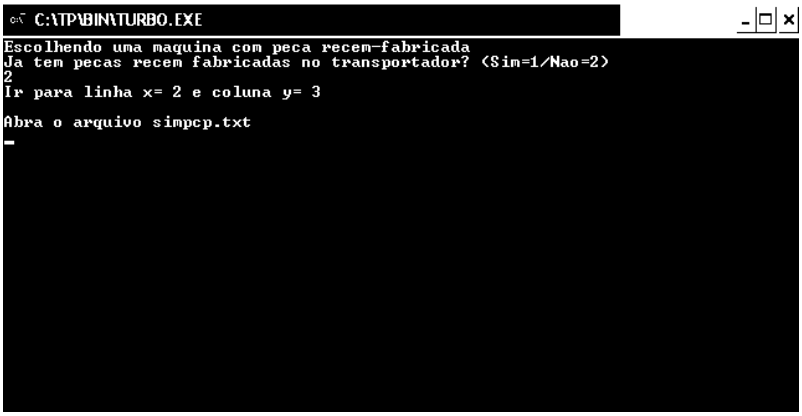

Figura 27: Exemplo de inserção de informações pelo funcionário 4.

```
1 1 3 1 10 1 0 2 0 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 0 5 0 0 0 0 0 0 0 0 0 0 0
2 1 1 0 0 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 10 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 3 4 32 46 0 0 1 2 1 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0
```

Figura 28: Ilustra atualização do banco de dados.

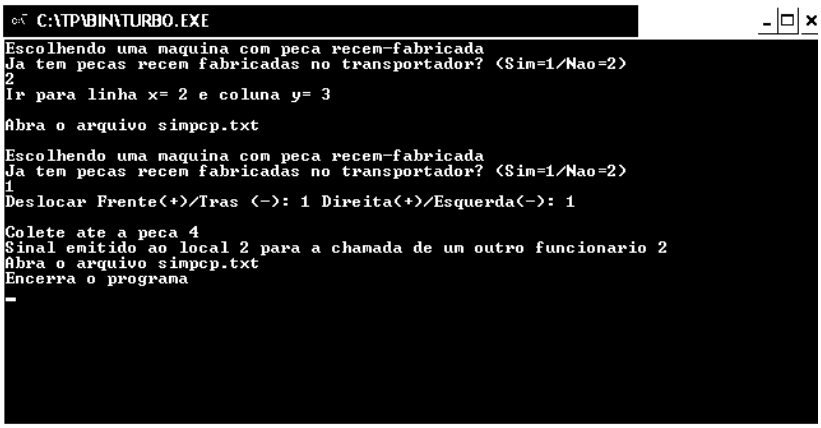

Figura 29: Exemplo de informação fornecido pelo sistema PP.

```
1 1 3 1 10 1 0 2 0 0 0 0 0 0 0 0 0 0 0
1 3 3 1 1 1 0 5 0 0 0 0 0 0 0 0 0 0 0
2 1 1 0 0 0 1 0 3 7 2 7 0 0 2 1 3 0 0
2 2 2 1 10 1 1 2 1 97 0 0 0 0 2 0 0 0 0
3 2 1 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 3 2 1 10 1 1 8 2 32 46 0 0 0 2 1 0 0 0
5 1 2 1 10 0 0 6 0 0 0 0 0 0 0 0 0 0 0
5 2 3 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5 3 1 1 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0
```

Figura 30: Ilustra atualização do banco de dados.

4 Simulação do tempo CPU consumido para testar os códigos e banco de dados propostos neste trabalho

Apesar de poder definir os códigos para acesso e atualização dos dados, é necessário agora testá-los para uma série de situações para verificar se existe um elevado tempo de acesso/atualização das informações do banco de dados, caso contrário, os códigos não são válidos para serem usados numa situação real. É necessário analisar quantas vezes cada tipo de funcionário acessa o computador e esta informação pode ser estimada a partir do(s) pedido(s). A Figura 31 mostra um esquema dos pedidos (uma forma de representação mais genérica da Tabela 01).

Peça tipo i	Operação 1 $_{\text{peça tipo i}}$ . . . . . .Operação n $_{\text{peça tipo i}}$
Peça tipo i+1	Operação 1 $_{\text{peça tipo i+1}}$ . . . . . .Operação n $_{\text{peça tipo i+1}}$
⋮	⋮ ⋮
Peça tipo n	Operação 1 $_{\text{peça tipo n}}$ . . . . . .Operação n $_{\text{peça tipo n}}$

Figura 31: Esquema genérico dos pedidos (peças com as suas respectivas operações).

É possível estimar o número de acessos por funcionário. Para o funcionário 1, será justamente n vezes. Assim, se houverem 50 tipos de peças no(s) pedido(s) (independentemente da separação dos pedidos por transportador), serão justamente 50 acessos. Para o funcionário 2, como toda vez que encerra uma determinada operação deve acessar o sistema, o número de acessos passa a ser em torno de $\sum_{i=1}^{n} Número\,operações\,peça\,tipo\,i$. Quanto ao funcionário 3, apenas para carregar e descarregar matéria-prima, ele acessa no máximo 3 vezes por cada operação da peça. O número de acessos para o funcionário 4 é mais complexo e não pode ser obtido simplesmente analisando o pedido. Isso porque quando este realiza a coleta, remove vários tipos de peças acabadas que se formam após a máquina ao mesmo tempo para melhor aproveitamento de viagem. Ou seja, um único acesso permite chegar até uma única máquina e coletar várias peças (e portanto várias caixas). Para simplificar, considerou-se que na pior das hipóteses, o funcionário 4 tenha que acessar n vezes. Só serão contabilizados os tempos quando o código acesso ao banco de dados, fora isso, tais como tempo de inserção das informações pelos funcionários não serão contabilizados. As Figuras 32, 33, 34 e 35 mostram a relação entre o tempo CPU de busca e fornecimento de informações por cada funcionário em relação ao tamanho do layout, onde número de operações representa o número de operações de acesso. Este tempo é o tempo total CPU para executar todo o pedido do dia. É claro que na prática, vários funcionários acessam o computador ao mesmo tempo, o que significa que o tempo de CPU é menor ainda.

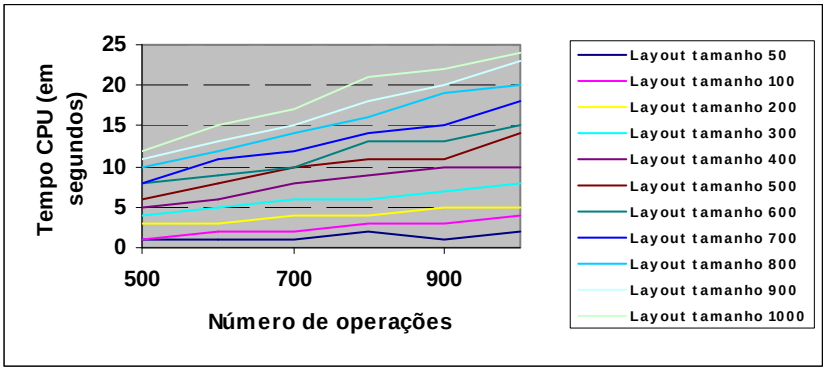

Figura 32: Tempo de CPU para o funcionário 1.

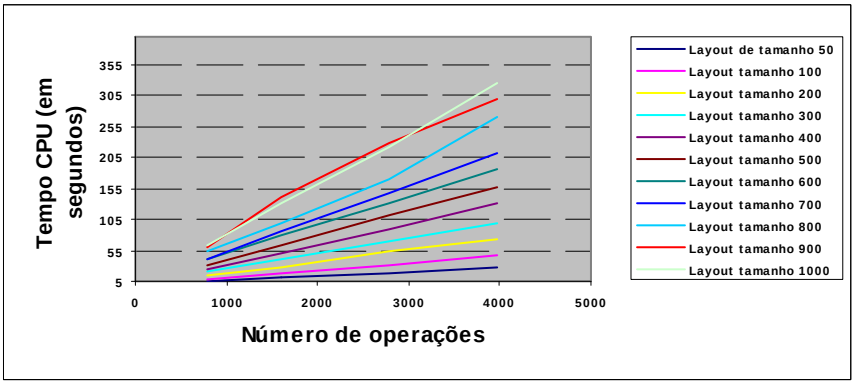

Figura 32: Tempo de CPU para o funcionário 2.

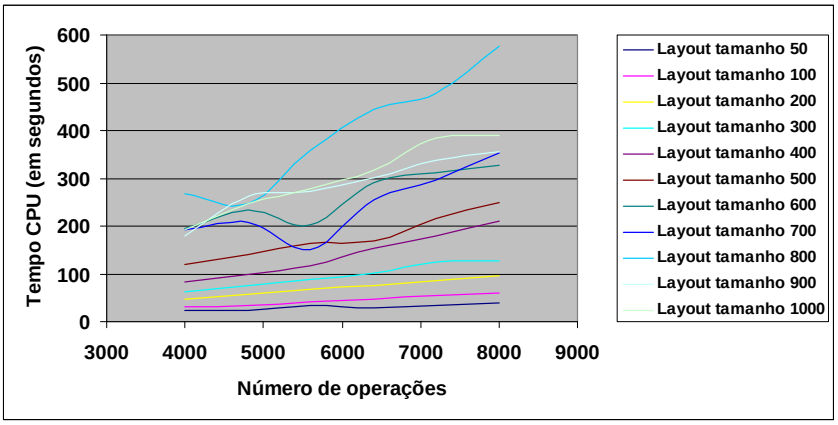

Figura 33: Tempo de CPU para o funcionário 3.

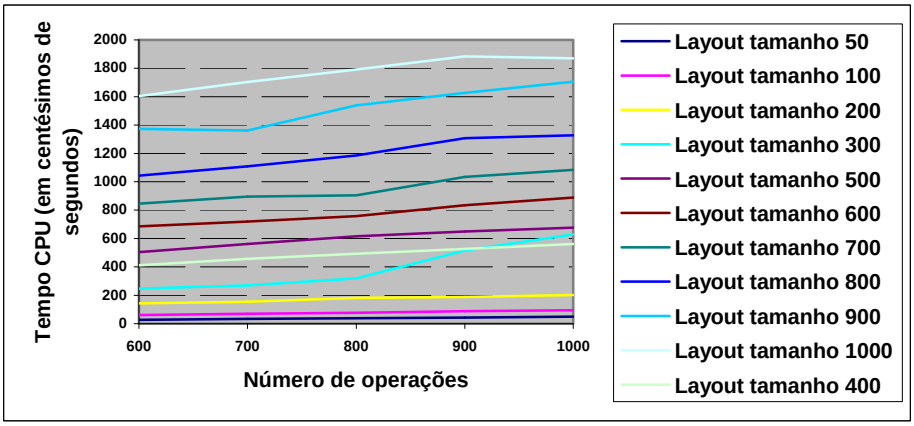

Figura 34: Tempo de CPU para o funcionário 4.

5 Conclusão

Um sistema completo de programação da produção pode ser, finalmente, implementado numa empresa real para a aplicação simultânea de critérios de seleção de recursos à medida que os pedidos chegam à fábrica, fato que anteriormente não era visto nos artigos da literatura.

Independentemente do tamanho do layout (considerando até tamanho de 1000 máquinas), os resultados mostram que os tempos de busca e atualização são praticamente desprezíveis, considerando um dia de trabalho (24h).

Visto que o tempo CPU é praticamente desprezível executando para todo pedido, não é necessário contabilizar este tempo às atividades referentes à cada funcionário e portanto não existe incremento do tempo total das peças.

Os resultados também mostram que aumentando tanto o número de operações quanto o tamanho do layout, os tempos de CPU em ambos aumentam quase que proporcionalmente.

Como proposta para futuros trabalhos, pode-se elaborar novas formas de inserção de informações pelos funcionários diferente deste trabalho. Isso porque por fazer uso de computadores, nem sempre a empresa pode ser capaz de adquirir tal tecnologia no início de suas atividades. Pode-se também focar em como armazenar as informações bem como a forma de acesso quando se trabalha com outros tipos de critérios não abordados neste trabalho.

Agradecimentos

Os autores agradecem à CAPES e CNPq pelo apoio financeiro para a realização deste trabalho.

Referências

BENJAAFAR, S. *Design of Flexible Layouts for Manufacturing Systems*. 95 Engineering Management Conference, 1995 IEEE.

BENJAAFAR, S.; Sheikhzadeh, M. *Design of Flexible Plant Layout, IEE Transactions*. Vol. 32, p. 309-322, 2000.

HAM, I.; Hitomi, K.; Yoshida, T. Group technology – Applications to production management. International Series in Management Science/Operations Research, The Pennsylvania State University, U.S.A, 1985.

JÚNIOR, J. H. C. G.; Filho, E. V. G. *Análise de Desempenho dos Arranjos Físicos Distribuídos Operando sob Roteamento de Peças com Flexibilidade de Sequenciamento, Revista Gestão Industrial*. Vol. 3, n. 1, p. 01-12, 2007.

LAHMAR, M.; Benjaafar, S. *Design of Distributed Layout, IIE Transactions*. Vol. 37, p. 303-318, 2005.

MONTREUIL, B.; Venkatadri, U.; Ratliff, H.D. *Generating a Layout from a Design Skeleton, IIE Transactions*. Vol. 25, n. 1, p. 3-15, 1993.

NETO, A. R. P.; Chin., S. Y.; Filho, E. V. G. *Design of Distributed Layouts with Operational Efficiency Considerations*. 19th International Conference on Production Research. Proceedings of 19th ICPR, Valparaiso, Chile, 2007.

Apêndice 1

```
Program simulando_banco_de_dados_PCP_Parte1;

uses crt;
var
i,j, total_maquinas, maq_seguinte, local1_x, local1_y,
menor1,contador, contador1, menor, escolhido_x, escolhido_y, entrega,
resposta, conta_entrega, caixas: integer;
maq_tipo, posicao_x, posicao_y, situacao, presenca, pecas_acabadas, fila_antes,
fila_depois, linha_contador, armazena_x, armazena_y, pecas_semi_acabadas: array[1..20] of integer;
tipos_peca_fila, lote: array[1..10, 1..10] of integer;
label 1;



{Ler arquivo externo}
Procedure ler_simpcp;
const
   filename='c:simpcp.txt';

var
  infile: text;


begin
   assign (infile, filename);
   reset(infile);
total_maquinas:=9;

for i:=1 to total_maquinas do
   begin
     read(infile, maq_tipo[i]);
     read(infile, posicao_x[i]);
     read(infile, posicao_y[i]);
     read(infile, situacao[i]);
     read(infile, presenca[i]);
     read(infile, pecas_semi_acabadas[i]);
     read(infile, pecas_acabadas[i]);
     read(infile, Fila_antes[i]);
     read(infile, Fila_depois[i]);
     for j:=1 to 5 do
       read(infile, Tipos_peca_fila[i,j]);
     for j:=1 to 5 do
       read(infile, lote[i,j]);
     readln(infile);
   end;
close(infile);
end;


Procedure imprima_simpcp;
const
   filename='c:simpcp.txt';

var
  outfile: text;


begin
```

```
    assign (outfile, filename);
    rewrite(outfile);
    total_maquinas:=9;

for i:=1 to total_maquinas do
    begin
      write(outfile, maq_tipo[i],' ');
      write(outfile, posicao_x[i],' ');
      write(outfile, posicao_y[i],' ');
      write(outfile, situacao[i],' ');
      write(outfile, presenca[i],' ');
      write(outfile, pecas_semi_acabadas[i],' ');
      write(outfile, pecas_acabadas[i],' ');
      write(outfile, Fila_antes[i],' ');
      write(outfile, Fila_depois[i],' ');
      for j:=1 to 5 do
        write(outfile, Tipos_peca_fila[i,j],' ');
      for j:=1 to 5 do
        write(outfile, lote[i,j],' ');
      writeln(outfile);
    end;
close(outfile);
end;


begin

local1_x:=4;
local1_y:=2;


clrscr;
writeln('*************Informacao inserida pelo funcionario 1************');
1:
writeln('Entre com o tipo de maquina da operacao seguinte');
readln(maq_seguinte);
writeln('E a primeira entrega? (1=sim/2=nao)');
readln(entrega);
writeln('Quantas caixas para aquela maquina estao sendo levadas? (para simplificar, adote 1)');
readln(caixas);
if entrega =1 then
  conta_entrega:=1
  else
  conta_entrega:=conta_entrega+1;

ler_simpcp;
contador:=0;
for i:=1 to total_maquinas do
  if (maq_tipo[i] = maq_seguinte) and (situacao[i] = 0) then
       begin
         contador:=contador+1;
         linha_contador[contador]:=i;
         armazena_x[contador]:=posicao_x[i];
         armazena_y[contador]:=posicao_y[i];
       end;

if contador<>0 then
 begin
  menor:=1000;
  for i:=1 to contador do
    begin
      menor1:=abs(local1_x-armazena_x[i])+abs(local1_y-armazena_y[i]);
      if menor1<=menor then
        begin
         menor:=menor1;
         contador1:=linha_contador[i];
         escolhido_x:=armazena_x[i];
         escolhido_y:=armazena_y[i];
        end;
    end;
  situacao[contador1]:=1;
  if conta_entrega =1 then
    writeln('Ir para linha: ',escolhido_x,' e coluna: ',escolhido_y)
  else
    writeln('Deslocar frente(+)/tras(-) :',escolhido_x-local1_x,
    ' direita(+)/esquerda(-) :',local1_y-escolhido_y);
  fila_antes[contador1]:=fila_antes[contador1]+caixas-1;
 end;

 if contador=0 then
   begin
     menor:=1000;
     for i:=1 to total_maquinas do
       if (maq_tipo[i]=maq_seguinte) and (fila_antes[i]<=menor) then
          begin
            menor:=fila_antes[i];
            contador1:=i;
            escolhido_x:=posicao_x[i];
            escolhido_y:=posicao_y[i];
          end;
     if conta_entrega =1 then
      writeln('Ir para linha: ',escolhido_x,' e coluna: ',escolhido_y)
     else
       begin
        writeln('O escolhido foi : x=',escolhido_x,' y=',escolhido_y);
        writeln('Deslocar frente(+)/tras(-) :',escolhido_x-local1_x,
        ' direita(+)/esquerda(-) :',local1_y-escolhido_y);
       end;
     fila_antes[contador1]:=fila_antes[contador1]+caixas;
   end;
local1_x:=escolhido_x;
local1_y:=escolhido_y;
imprima_simpcp;
writeln('Abra o arquivo simpcp.txt');
readln;
writeln('Todas as materias-primas entregues? (1=sim/2=nao)');
readln(resposta);
if resposta = 2 then
```

```
    goto 1 else writeln('Retorne ao Local 1');

  readln;
end.
```

Apêndice 2

```
  Program simulando_banco_de_dados_PCP_Parte2;

  uses crt;
  var
  tipos_maq, total_maquinas, i, j, x, y, linha_escolhida,
  contador, fila_antes1, ultima_peca: integer;
  maq, maq_tipo, posicao_x, posicao_y, situacao, presenca, pecas_acabadas, fila_antes,
  fila_depois, ocupado, pecas_acabadas1, pecas_semi_acabadas,
  pecas_semi_acabadas1: array[1..20] of integer;
  tipos_peca_fila, lote: array[1..10, 1..10] of integer;

  {Ler arquivo externo}
  Procedure ler_simpcp;
  const
     filename='c:simpcp.txt';

  var
    infile: text;


  begin
     assign (infile, filename);
     reset(infile);
     total_maquinas:=9;

  for i:=1 to total_maquinas do
     begin
       read(infile, maq_tipo[i]);
       read(infile, posicao_x[i]);
       read(infile, posicao_y[i]);
       read(infile, situacao[i]);
       read(infile, presenca[i]);
       read(infile, pecas_semi_acabadas[i]);
       read(infile, pecas_acabadas[i]);
       read(infile, Fila_antes[i]);
       read(infile, Fila_depois[i]);
       for j:=1 to 5 do
          read(infile, Tipos_peca_fila[i,j]);
       for j:=1 to 5 do
          read(infile, lote[i,j]);
       readln(infile);
     end;
  close(infile);
  end;


  Procedure imprima_simpcp;
  const
     filename='c:simpcp.txt';

  var
    outfile: text;


  begin
     assign (outfile, filename);
     rewrite(outfile);
     total_maquinas:=9;

  for i:=1 to total_maquinas do
     begin
       write(outfile, maq_tipo[i],' ');
       write(outfile, posicao_x[i],' ');
       write(outfile, posicao_y[i],' ');
       write(outfile, situacao[i],' ');
       write(outfile, presenca[i],' ');
       write(outfile, pecas_semi_acabadas[i],' ');
       write(outfile, pecas_acabadas[i],' ');
       write(outfile, Fila_antes[i],' ');
       write(outfile, Fila_depois[i],' ');
       for j:=1 to 5 do
          write(outfile, Tipos_peca_fila[i,j],' ');
       for j:=1 to 5 do
          write(outfile, lote[i,j],' ');
       writeln(outfile);
     end;
  close(outfile);
  end;


  begin
  clrscr;
  writeln('*************Informacao inserida pelo funcionario 2************');
  writeln('Entre com o local em x onde se encontra o funcionario 2');
  readln(x);
  writeln('Entre tambem com o local em y onde se encontra o funcionario 2');
  readln(y);
  writeln('A peca atual  na maquina e a ultima peca? (Sim=1/Nao=2)');
  readln(ultima_peca);
  if ultima_peca=2 then
    begin
      writeln('Entre com a fila (tamanho da fila antes) apos a montagem da peca na maquina');
      readln(fila_antes1);
    end
    else
    fila_antes1:=0;
  ler_simpcp;
  for i:=1 to total_maquinas do
     if (posicao_x[i]=x) and (posicao_y[i]=y) then
       linha_escolhida:=i;
```

```
fila_antes[linha_escolhida]:=fila_antes1;

if (fila_antes[linha_escolhida]=0) then
  begin
    writeln('O funcionario 2 esta ocupado? (Sim=1/Nao=2)');
    readln(ocupado[linha_escolhida]);
    if ocupado[linha_escolhida] =1 then
        situacao[linha_escolhida]:=1;
    if ocupado[linha_escolhida]=2 then
        situacao[linha_escolhida]:=0;
  end
  else
    situacao[linha_escolhida]:=1;

imprima_simpcp;
writeln('Abra o arquivo simpcp.txt');
readln;
ler_simpcp;
writeln('Existem pecas acabadas? (Sim=1/Nao=2)');
readln(pecas_acabadas1[linha_escolhida]);
if pecas_acabadas1[linha_escolhida]=1 then
  begin
    pecas_acabadas[linha_escolhida]:=1;
    fila_depois[linha_escolhida]:=fila_depois[linha_escolhida]+1;
    writeln('Entre com o tipo de peca-acabada em fila');
    readln(Tipos_peca_fila[linha_escolhida, fila_depois[linha_escolhida]]);
    writeln('Entre com o seu tamanho do lote (unids.)');
    readln(lote[linha_escolhida, fila_depois[linha_escolhida]]);
  end
  else
  begin
    pecas_acabadas[linha_escolhida]:=0;
    fila_depois[linha_escolhida]:=0;
    tipos_peca_fila[linha_escolhida,1]:=0;
    lote[linha_escolhida,1]:=0;
  end;

imprima_simpcp;
writeln('Abra o arquivo simpcp.txt');
readln;
ler_simpcp;
writeln('Existe fila de pecas semi-acabadas que se forma depois da maquina? (Sim=1/ Nao=2)');
readln(pecas_semi_acabadas1[linha_escolhida]);
if pecas_semi_acabadas1[linha_escolhida]=1 then
  pecas_semi_acabadas[linha_escolhida]:=1;
if pecas_semi_acabadas1[linha_escolhida]=2 then
  pecas_semi_acabadas[linha_escolhida]:=0;
imprima_simpcp;
writeln('Abra o arquivo simpcp.txt');
readln;
end.
```

Apêndice 3

```
Program simulando_banco_de_dados_PCP_Parte3;

uses crt;
var
tipos_maq, total_maquinas, i, j, x, y, linha_escolhida1,contador, menor, menor1,
linha_escolhida, escolhido_x, escolhido_y, carregado, maquina, linha_escolhida4: integer;
maq, maq_tipo, posicao_x, posicao_y, situacao, presenca, pecas_acabadas, fila_antes,
fila_depois,pecas_semi_acabadas, armazena_x, armazena_y, linha_escolhida3: array[1..20] of integer;
tipos_peca_fila, lote: array[1..10, 1..10] of integer;
label 1,2, 3;


{Ler arquivo externo}
Procedure ler_simpcp;
const
    filename='c:simpcp.txt';

var
  infile: text;


begin
    assign (infile, filename);
    reset(infile);
total_maquinas:=9;

for i:=1 to total_maquinas do
    begin
      read(infile, maq_tipo[i]);
      read(infile, posicao_x[i]);
      read(infile, posicao_y[i]);
      read(infile, situacao[i]);
      read(infile, presenca[i]);
      read(infile, pecas_semi_acabadas[i]);
      read(infile, pecas_acabadas[i]);
      read(infile, Fila_antes[i]);
      read(infile, Fila_depois[i]);
      for j:=1 to 5 do
        read(infile, Tipos_peca_fila[i,j]);
      for j:=1 to 5 do
        read(infile, lote[i,j]);
      readln(infile);
    end;
close(infile);
end;


Procedure imprima_simpcp;
const
    filename='c:simpcp.txt';
```

```
var
  outfile: text;


begin
    assign (outfile, filename);
    rewrite(outfile);
    total_maquinas:=9;

for i:=1 to total_maquinas do
    begin
      write(outfile, maq_tipo[i],' ');
      write(outfile, posicao_x[i],' ');
      write(outfile, posicao_y[i],' ');
      write(outfile, situacao[i],' ');
      write(outfile, presenca[i],' ');
      write(outfile, pecas_semi_acabadas[i],' ');
      write(outfile, pecas_acabadas[i],' ');
      write(outfile, Fila_antes[i],' ');
      write(outfile, Fila_depois[i],' ');
      for j:=1 to 5 do
        write(outfile, Tipos_peca_fila[i,j],' ');
      for j:=1 to 5 do
        write(outfile, lote[i,j],' ');
      writeln(outfile);
    end;
close(outfile);
end;

begin
3:
clrscr;
{Quando um dos funcionarios 3 estiver livre entao o sistema e acessado}
writeln('*************Informacao inserida pelo funcionario 3***********');
writeln('Insira a posicao do transportador 3 em x');
readln(x);
writeln('Insira a posicao do transportador 3 em y');
readln(y);
1:
{Informar ao PCP se este ja carregou ou nao}
writeln('O transportador 3 esta carregado? (Sim=1/Nao=2)');
readln(carregado);

if carregado=1 then
  goto 2;

ler_simpcp;

{Encontrar que linha da tabela o funcionario 3 se encontra}
for i:=1 to total_maquinas do
  if (x=posicao_x[i]) and (y=posicao_y[i]) then
    linha_escolhida1:=i;

presenca[linha_escolhida1]:=0;
imprima_simpcp;
ler_simpcp;
writeln('Veja o arquivo simpcp.txt');
readln;

{Faz uma busca por todos os locais por onde existem pecas semi-acabadas}
contador:=0;
for i:=1 to total_maquinas do
  if pecas_semi_acabadas[i]=1 then
    begin
      contador:=contador+1;
      armazena_x[contador]:=posicao_x[i];
      armazena_y[contador]:=posicao_y[i];
      linha_escolhida3[contador]:=i;
    end;

{Como existem pecas semi-acabadas, ja definir o transportador 3
como ausente no local de origem}
if contador<>0 then
 presenca[linha_escolhida1]:=10
 else goto 3;

imprima_simpcp;
ler_simpcp;
writeln('Veja o arquivo simpcp.txt');
readln;
{Encontrar o local mais proximo do x e y fornecido pelo funcionario 3}
menor:=1000;
for i:=1 to contador do
  begin
    menor1:=abs(armazena_x[i]-x)+abs(armazena_y[i]-y);
    if menor1<=menor then
      begin
        menor:=menor1;
        linha_escolhida:=linha_escolhida3[i];
        escolhido_x:=armazena_x[i];
        escolhido_y:=armazena_y[i];
      end;
  end;

{Informar ao funcionario 3 onde ele deve chegar para carregar as pecas semi-acabadas}
writeln('Deslocar Frente(+)/Tras(-) ',escolhido_x-x,
 ' Direita(+)/Esquerda(-) ',y-escolhido_y);

{Atualizar o transportador 3 (agora este esta considerado como carregado/ocupado)}
{Esta etapa e importante para garantir que o funcionario 3 nao seja escolhido
pelo sistema PCP, mesmo nao ter tido carregado a peca semi-acabada}
presenca[linha_escolhida]:=1;
imprima_simpcp;
writeln('Veja o arquivo simpcp.txt');
readln;
goto 1;
2:
```

```
{apos o carregamento, deve informar que tipo de operacao que a peca semi-acabada necessita}
writeln('Entre com o tipo de maquina da operacao seguinte');
readln(maquina);
ler_simpcp;

{Definindo para onde o funcionario 3 deve ir, verificando se existe alguma maquina livre}
contador:=0;
for i:=1 to total_maquinas do
  if (situacao[i]=0) and (maq_tipo[i]=maquina) then
     begin
       contador:=contador+1;
       armazena_x[contador]:=posicao_x[i];
       armazena_y[contador]:=posicao_y[i];
     end;


{Analisando qual das livres e a mais proxima}
if contador<>0 then
begin
 menor:=1000;
 for i:=1 to contador do
   begin
     menor1:=abs(armazena_x[i]-x)+abs(armazena_y[i]-y);
     if menor1<=menor then
       begin
         menor:=menor1;
         escolhido_x:=armazena_x[i];
         escolhido_y:=armazena_y[i];
       end;
   end;

 {Informando ao funcionario 3 onde este deve se deslocar}
 writeln('Deslocar Frente(+)/Tras(-) ',escolhido_x-x,
 ' Direita(+)/esquerda(-) ',y-escolhido_y);

 {atualizando automaticamente a informacao sobre o transportador estar ocupado ou nao}
 for i:=1 to total_maquinas do
   if (posicao_x[i]=escolhido_x) and (posicao_y[i]=escolhido_y) then
     linha_escolhida4:=i;
 presenca[linha_escolhida4]:=1;
 presenca[linha_escolhida]:=10;
 situacao[linha_escolhida4]:=1;
 imprima_simpcp;
 writeln('Veja o arquivo simpcp.txt');
end;


{fazendo a busca pela maquina de menor fila}
if contador=0 then
 begin
  menor:=1000;
  for i:=1 to total_maquinas do
    if (maq_tipo[i]=maquina) and (fila_antes[i]<=menor) then
      begin
       menor:=fila_antes[i];
       escolhido_x:=posicao_x[i];
       escolhido_y:=posicao_y[i];
      end;
  {atualizando automaticamente a informacao sobre o transportador estar ocupado ou nao}
  for i:=1 to total_maquinas do
    if (posicao_x[i]=escolhido_x) and (posicao_y[i]=escolhido_y) then
      linha_escolhida4:=i;
  presenca[linha_escolhida4]:=1;
  presenca[linha_escolhida]:=10;
  imprima_simpcp;
  writeln('Veja o arquivo simpcp.txt');
 end;
end.
```

Apêndice 4

```
Program simulando_banco_de_dados_PCP_Parte4;

uses crt;
var
tipos_maq, total_maquinas, i, j, local2_x, local2_y, meio_cheio,
contador, maior, linha_escolhida, escolhido_x, escolhido_y,
contador1, carregado, maquina, menor, menor1,
carregou: integer;
carga_transportador: real;
maq, maq_tipo, posicao_x, posicao_y, situacao, presenca, pecas_acabadas, fila_antes,
fila_depois,pecas_semi_acabadas, armazena_x, armazena_y, armazena1_x,
armazena1_y, linha_escolhida1, tamanho_fila, linha_escolhida3: array[1..20] of integer;
tipos_peca_fila, lote: array[1..10, 1..10] of integer;
peso: array[1..100] of real;
label 1,2, 3;


{Ler arquivo externo}
Procedure ler_simpcp;
const
   filename='c:simpcp.txt';

var
  infile: text;


begin
   assign (infile, filename);
   reset(infile);
total_maquinas:=9;

for i:=1 to total_maquinas do
   begin
     read(infile, maq_tipo[i]);
     read(infile, posicao_x[i]);
     read(infile, posicao_y[i]);
```

```
      read(infile, situacao[i]);
      read(infile, presenca[i]);
      read(infile, pecas_semi_acabadas[i]);
      read(infile, pecas_acabadas[i]);
      read(infile, Fila_antes[i]);
      read(infile, Fila_depois[i]);
      for j:=1 to 5 do
        read(infile, Tipos_peca_fila[i,j]);
      for j:=1 to 5 do
        read(infile, lote[i,j]);
      readln(infile);
    end;
close(infile);
end;


Procedure imprima_simpcp;
const
    filename='c:simpcp.txt';

var
  outfile: text;


begin
    assign (outfile, filename);
    rewrite(outfile);
    total_maquinas:=9;

for i:=1 to total_maquinas do
    begin
      write(outfile, maq_tipo[i],' ');
      write(outfile, posicao_x[i],' ');
      write(outfile, posicao_y[i],' ');
      write(outfile, situacao[i],' ');
      write(outfile, presenca[i],' ');
      write(outfile, pecas_semi_acabadas[i],' ');
      write(outfile, pecas_acabadas[i],' ');
      write(outfile, Fila_antes[i],' ');
      write(outfile, Fila_depois[i],' ');
      for j:=1 to 5 do
        write(outfile, Tipos_peca_fila[i,j],' ');
      for j:=1 to 5 do
        write(outfile, lote[i,j],' ');
      writeln(outfile);
    end;
close(outfile);
end;


procedure ler_peso;
const
    filename='c:peso1.txt';

var
  infile: text;
  tipos_peca, j: integer;

begin
    assign (infile, filename);
    reset(infile);
    tipos_peca:=100;
    readln(infile);

j:=2;
for i:=1 to tipos_peca do
    begin
      read(infile,j);
      readln(infile,peso[i]);
    end;

close(infile);
end;

begin
clrscr;
ler_peso;
local2_x:=0;
local2_y:=2;
meio_cheio:=0;
writeln('Escolhendo uma maquina com peca recem-fabricada');
{Ao receber o sinal do sistema PCP dizendo que existem pecas acabadas,
o funcionario 4 acessa o sistema}
{O sistema deve checar quais as maquinas que possuem pecas acabadas}
2:
writeln('Ja tem pecas recem fabricadas no transportador? (Sim=1/Nao=2)');
readln(carregou);
ler_simpcp;
contador:=0;
for i:=1 to total_maquinas do
  if fila_depois[i] <>0 then
    begin
      contador:=contador+1;
      linha_escolhida1[contador]:=i;
      tamanho_fila[contador]:=fila_depois[i];
      armazena_x[contador]:=posicao_x[i];
      armazena_y[contador]:=posicao_y[i];
    end;

if contador=0 then goto 3;
{Analisando a de maior fila}
maior:=0;
for i:=1 to contador do
 if tamanho_fila[i]>maior then
   begin
     maior:=tamanho_fila[i];
     linha_escolhida:=linha_escolhida1[i];
     escolhido_x:=armazena_x[i];
```

```
      escolhido_y:=armazena_y[i];
    end;

{analisando se existe o mesmo tamanho de maior fila e a mais proxima}
contador1:=0;
for i:=1 to total_maquinas do
 if fila_depois[i]=maior then
   begin
     contador1:=contador1+1;
     linha_escolhida3[contador1]:=i;
     armazena1_x[contador1]:=posicao_x[i];
     armazena1_y[contador1]:=posicao_y[i];
   end;

{So le esta parte se houverem duas filas do mesmo tamanho e escolhe a mais proxima}
if contador1>=2 then
 begin
   menor:=1000;
   for i:=1 to contador1 do
     begin
       menor1:=abs(armazena1_x[i]-local2_x)+abs(armazena1_y[i]-local2_y);
       if menor1<=menor then
         begin
           menor:=menor1;
           linha_escolhida:=linha_escolhida3[i];
           escolhido_x:=armazena1_x[i];
           escolhido_y:=armazena1_y[i];
         end;
     end;
 end;


if carregou=2 then
 {Deslocando do local 2 ate a maquina de maior fila de pecas recem-fabricadas}
 writeln('Ir para linha x= ',escolhido_x,
 ' e coluna y= ',escolhido_y)
 else
 {Deslocando do local 2 ate a maquina de maior fila de pecas recem-fabricadas}
 writeln('Deslocar Frente(+)/Tras (-): ',escolhido_x-local2_x,
 ' Direita(+)/Esquerda(-): ',local2_y-escolhido_y);
readln;

if meio_cheio= 0 then
 carga_transportador:=0;


For i:=1 to maior do
 begin
   carga_transportador:=carga_transportador+peso[tipos_peca_fila[linha_escolhida,i]]*
   lote[linha_escolhida, i];
   fila_depois[linha_escolhida]:=fila_depois[linha_escolhida]-1;
   if carga_transportador>=20 then
     begin
       writeln('Colete ate a peca ',tipos_peca_fila[linha_escolhida,i-1]);
       fila_depois[linha_escolhida]:=fila_depois[linha_escolhida]+1;
       writeln('Sinal emitido ao local 2 para a chamada de um outro funcionario 2');
       contador:=0;
       for i:=i to maior do
          begin
            contador:=contador+1;
            tipos_peca_fila[linha_escolhida, contador]:=tipos_peca_fila[linha_escolhida, i];
            lote[linha_escolhida, contador]:=lote[linha_escolhida, i];
          end;

       for j:=contador+1 to 5 do
          begin
            tipos_peca_fila[linha_escolhida, j]:=0;
            lote[linha_escolhida, j]:=0;
          end;

       imprima_simpcp;
       ler_simpcp;
       writeln('Abra o arquivo simpcp.txt');
       writeln('Encerra o programa');
       readln;
       goto 3;
     end;
 end;

if carga_transportador<20 then
 begin
   pecas_acabadas[linha_escolhida]:=0;
   fila_depois[linha_escolhida]:=0;
   for j:=1 to 5 do
     begin
       tipos_peca_fila[linha_escolhida,j]:=0;
       lote[linha_escolhida,j]:=0;
     end;
   imprima_simpcp;
   writeln('Abra o arquivo simpcp.txt');
   readln;
   local2_x:=escolhido_x;
   local2_y:=escolhido_y;
   meio_cheio:=meio_cheio+1;
   writeln('Escolhendo uma maquina com peca recem-fabricada');
   goto 2;
 end;
3:
end.
```