5
Método Heurístico Proposto

A idéia do método a ser desenvolvido neste estudo é adaptar o método de Newton, proposto por Fliege *et al.* (2008) e aplicado para resolver problemas de otimização multiobjetivo não linear com variáveis contínuas, para resolver problemas de escalonamento em vários ambientes. Nessa adaptação, tem-se que considerar problemas com variáveis inteiras ou binárias, e ainda considerar um novo critério de parada. Observa-se que o método original utiliza como teste de parada a satisfação da condição necessária de otimalidade de primeira ordem, ou seja, gradiente da função objetivo nulo em um iterado, que agora não pode ser mais usado já que a noção de gradiente não se aplica em variáveis inteiras. A seguir, são apresentados as variações do método de Newton para problemas de escalonamento nos ambientes FSP, fJSP, iRS/OS e APS.

5.1
Método proposto para o FSP

O algoritmo correspondente ao método proposto para FSP é iterativo. Ele visa melhorar a seqüência de tarefas a cada iteração, obtendo, por fim, um conjunto de soluções *não-dominadas* em relação às medidas de desempenho sendo consideradas simultaneamente.

5.1.1
Estrutura principal

O algoritmo parte de uma solução inicial, quer dizer, uma seqüência s^* gerada em cada iteração. Esta solução melhora primeiro mediante a vizinhança gerada pelo método de inserção (descrita na seção 2.1.4-3) e depois a vizinhança gerada pelo método de troca de duas tarefas (descrita na seção 2.1.4-2). Quando possível, gera uma nova seqüência de operações em que pelo menos uma das funções objetivo é melhorada. Este enfoque de melhoria misturando os dois tipos de vizinhança foi proposto por Ho (1993). O algoritmo segue enquanto o parâmetro "melhoria" for verdadeiro. O pseudocódigo da estrutura principal do algoritmo proposto é descrito a seguir.

Procedimento: Estrutura principal - FSP
Input: N_{iter};
Output: ND;
Início
  $S \leftarrow \emptyset, ND \leftarrow \emptyset$;
  para $i \leftarrow 1$ **até** N_{iter} **faça**
    Gere uma solução inicial s^*;
    melhoria ← verdadeiro;
    enquanto (melhoria = verdadeiro) **faça**
      Melhore a seqüência de tarefas s^* mediante o método de inserção;
      $s' \leftarrow s^*$;
      Melhore a seqüência de tarefas s^*mediante o método de troca;
      se ($s' = s^*$) **então**
        melhoria ← falso;
      fim se;
    fim enquanto;
    $S \leftarrow S \cup \{s^*\}$;
  fim para;
  $ND \leftarrow$ Soluções não dominadas de S;
Fim

Na estrutura principal do algoritmo proposto para o FSP, N_{iter} é o número de iterações ou número de soluções geradas inicialmente e que serão melhoradas; se s^* não muda ($s' = s^*$) significa que os objetivos não melhoraram; S é o conjunto de soluções melhoradas de todas as seqüências inicialmente geradas; ND é o conjunto de soluções *não-dominadas* do conjunto S. A seguir, são detalhados os componentes ou subrotinas do algoritmo proposto.

5.1.2 Melhorar seqüência de tarefas bi-objetivo

O procedimento parte de uma seqüência inicial de tarefas $s_0 = s^*$, para $k = 0$. Seu objetivo é melhorar em cada iteração pelo menos uma função objetivo sem piorar outra função. Para isso, em cada iteração k gera-se a vizinhança $N(s_k)$ da seqüência s_k e testando o parâmetro θ escolhe-se o melhor vizinho de $N(s_k)$, de modo que minimize pelo menos um objetivo. O melhor vizinho de $N(s_k)$ será s_{k+1} e k passa a ser $k + 1$. O procedimento continuará enquanto for possível reduzir pelo menos uma função objetivo (caso em que descida=verdadeiro). Ao término do procedimento a melhor seqüência encontrada s_k será atribuída a s^*. O esquema geral do procedimento é descrito a seguir.

Procedimento: Newton adaptado para o FSP bi-objetivo

Input: s^*;

Output: s^*;

Início

$s_0 \leftarrow s^*, k \leftarrow 0,$ descida $\leftarrow$ verdadeiro;

enquanto (descida = verdadeiro) **faça**

$df_1 \leftarrow \emptyset, df_2 \leftarrow \emptyset;$

$\theta \leftarrow \min\limits_{s \in N(s_k)} \max\limits_{j=1,2}(f_j(s) - f_j(s_k));$

se $(\theta < 0)$ **então** $s_{k+1} \leftarrow \underset{s \in N(s_k)}{argmin} \max\limits_{j=1,2}(f_j(s) - f_j(s_k));$

senão se $(\theta = 0)$ **então**

para cada $s \in N_{(s_k)}$ **faça**

se ($\max\limits_{j=1,2}(f_j(s) - f_j(s_k)) = 0$) **então**

$df_1 \leftarrow df_1 \cup \{f_1(s) - f_1(s_k)\}, df_2 \leftarrow df_2 \cup \{f_2(s) - f_2(s_k)\};$

senão

$df_1 \leftarrow df_1 \cup \{0\}, df_2 \leftarrow df_2 \cup \{0\};$

fim se;

fim para cada;

se ($\min\limits_{s \in N(s_k)} \min\limits_{j=1,2}(df_j(s)) = 0$) **então** descida $\leftarrow$ falso;

senão se ($\min\limits_{s \in N(s_k)} df_1(s) = 0$) **então** $s_{k+1} \leftarrow \underset{s \in N(s_k)}{argmin}\ df_2(s);$

senão se ($\min_{s \in N(s_k)} df_2(s) = 0$) **então** $s_{k+1} \leftarrow \underset{s \in N(s_k)}{argmin}\ df_1(s)$;

senão $s_{k+1} \leftarrow \underset{s \in N(s_k)}{argmin}\, df_1(s)$;

fim se;

senão descida ← falso;

fim se;

se (descida=verdadeiro) **então** $k \leftarrow k + 1$;

fim se;

fim enquanto;

$s^* \leftarrow s_k$;

Fim

No procedimento, s denota cada vizinho de $s_k \in N(s_k)$ na iteração k; f_1 e f_2 são as funções objetivo a minimizar; θ mede a variação de f_1 e f_2 na solução corrente s_k e nas soluções vizinhas s. Se θ for negativo, significa que é possível reduzir simultaneamente f_1 e f_2. Se θ for positivo, então, pelo menos uma função objetivo piora, fazendo com que o procedimento termine. Se θ for zero, então, é possível melhorar no máximo um dos objetivos sem piorar o outro ou simplesmente nenhum objetivo melhora.

Aqui, df_1 e df_2 são os conjuntos das variações não positivas de f_1 e f_2 respectivamente.

Após a determinação de df_1 e df_2, verifica-se se $\min_{s \in N(s_k)} \min_{j=1,2} df_j(s)$ é zero, ou seja, verifica-se a situação em que, para todo $s \in N(s_k)$, não é possível continuar melhorando nenhum objetivo. No caso em que o valor mínimo é diferente de zero, tem-se a situação em que é possível melhorar um ou dois objetivos sem piorar o outro objetivo.

Se $\min_{s \in N(s_k)} df_1(s)$ for zero, então, f_2 pode ser melhorado e f_1 é inalterável. Se $\min_{s \in N(s_k)} df_2(s)$ for zero, então, f_1 pode ser melhorado e f_2 é inalterável. Senão, é possível melhorar qualquer objetivo sem piorar o outro (existem duas direções de busca). Nesse caso, deve-se optar por seguir uma das direções com a qual se obtém melhores resultados; em nosso caso optamos por melhorar f_1.

5.2
Método proposto para o fJSP, iRS/OS e APS

O algoritmo correspondente ao método proposto para fJSP, iRS/OS e APS é iterativo. Ele visa melhorar a alocação de máquinas em primeiro lugar, depois a seqüência de operações, obtendo, por fim, um conjunto de soluções *não-dominadas* em relação às medidas de desempenho sendo consideradas simultaneamente.

5.2.1
Estrutura principal

O algoritmo parte uma solução inicial, quer dizer, uma seqüência de operações e uma alocação de máquinas (s^*, v^*) gerada em cada iteração, que seja factível em relação à precedência das operações e à capacidade disponível das máquinas, no caso dos problemas iRS/OS e APS (ZHANG *ET AL.*, 2006a). O algoritmo segue enquanto o parâmetro "melhoria" for verdadeiro. O pseudocódigo da estrutura principal do algoritmo proposto é descrito a seguir.

Procedimento: Estrutura principal – fJSP, iRS/OS e APS

Input: N_{iter};

Output: ND;

Início

  $S \leftarrow \emptyset, ND \leftarrow \emptyset$;

  para $i \leftarrow 1$ **até** N_{iter} **faça**

    Gere uma solução inicial (s^*, v^*);

    melhoria ← verdadeiro;

    enquanto (melhoria = verdadeiro) **faça**

      $v' \leftarrow v^*$;

      Melhore a alocação de máquinas v^*;

      se $(v' = v^*)$ **então**

        melhoria ← falso;

      senão

        Melhore a seqüência s^*;

fim se;

fim enquanto;

$S \leftarrow S \cup \{s^*, v^*\}$;

fim para;

$ND \leftarrow$ Soluções não dominadas de S;

Fim

Na estrutura principal do algoritmo proposto para problemas fJSP, iRS/OS e APS, N_{iter} é o número de iterações ou número de soluções geradas inicialmente e que serão melhoradas; se v^* não muda ($v' = v^*$), significa que os objetivos não melhoraram; S é o conjunto de soluções melhoradas de todas as soluções inicialmente geradas; ND é o conjunto de soluções *não-dominadas* do conjunto S. A seguir, são detalhados os componentes ou subrotinas do algoritmo proposto.

5.2.2
Melhorar alocação de máquinas

Considerando s^* fixa, o procedimento parte de uma alocação inicial de máquinas $v_0 = v^*$, para $k = 0$. Seu objetivo é melhorar em cada iteração pelo menos uma função objetivo sem piorar outra função. Para isso, em cada iteração k gera-se a vizinhança $M(v_k)$ da alocação v_k e testando o parâmetro θ escolhe-se o melhor vizinho de $M(v_k)$ de modo que minimize pelo menos um objetivo. O melhor vizinho de $M(v_k)$ será v_{k+1} e k passa a ser $k + 1$. O procedimento continuará enquanto for possível reduzir pelo menos uma função objetivo (caso em que descida=verdadeiro). Ao término do procedimento a melhor alocação encontrada v_k será atribuída a v^*. A vizinhança $M(v_k)$ é gerada pelo método descrito na seção 2.2.4 para o fJSP e pelo mesmo método descrito nas seções 2.3.4 – 2.4.4 no caso dos problemas iRS/OS e APS respectivamente.

5.2.2.1 Melhorar alocação de máquinas bi-objetivo

O esquema geral do procedimento para melhorar alocação de máquinas para os problemas fJSP, iRS/OS e APS, quando se considera simultaneamente duas medidas de desempenho, é apresentado a seguir.

Procedimento: Newton adaptado para o fJSP, iRS/OS e APS bi-objetivo

Input: s^*, v^*;

Output: v^*;

Início

$v_0 \leftarrow v^*, k \leftarrow 0,$ descida ← verdadeiro;

enquanto (descida = verdadeiro) **faça**

$df_1 \leftarrow \emptyset, df_2 \leftarrow \emptyset$;

$\theta \leftarrow \min\limits_{v \in M(v_k)} \max\limits_{j=1,2}(f_j(s^*, v) - f_j(s^*, v_k))$;

se $(\theta < 0)$ **então** $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin} \max\limits_{j=1,2}(f_j(s^*, v) - f_j(s^*, v_k))$;

senão se $(\theta = 0)$ **então**

para cada $v \in M_{(v_k)}$ **faça**

se ($\max\limits_{j=1,2}(f_j(s^*, v) - f_j(s^*, v_k)) = 0$) **então**

$df_1 \leftarrow df_1 \cup \{f_1(s^*, v) - f_1(s^*, v_k)\}$;

$df_2 \leftarrow df_2 \cup \{f_2(s^*, v) - f_2(s^*, v_k)\}$;

senão

$df_1 \leftarrow df_1 \cup \{0\}, df_2 \leftarrow df_2 \cup \{0\}$;

fim se;

fim para cada;

se ($\min\limits_{v \in M(v_k)} \min\limits_{j=1,2}(df_j(v)) = 0$) **então** descida ← falso;

senão se ($\min\limits_{v \in M(v_k)} df_1(v) = 0$) **então** $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin}\, df_2(v)$;

senão se ($\min\limits_{v \in M(v_k)} df_2(v) = 0$) **então** $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin}\, df_1(v)$;

senão $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin}\, df_1(v)$;

fim se;

senão descida ← falso;

fim se;

se (descida=verdadeiro) **então** $k \leftarrow k + 1$;

fim se;

fim enquanto;

$v^* \leftarrow v_k$;

Fim

No procedimento, v denota cada vizinho de $v_k \in M(v_k)$ na iteração k; f_1 e f_2 são as funções objetivo a minimizar; θ mede a variação de f_1 e f_2 da solução corrente (s^*, v_k) e as soluções vizinhas (s^*, v). Se θ for negativo, significa que é possível reduzir simultaneamente f_1 e f_2. Se θ for positivo, então, pelo menos uma função objetivo piora, o qual faz que o procedimento termine. Se θ for zero, então é possível melhorar no máximo um dos objetivos sem piorar o outro ou simplesmente nenhum objetivo melhora.

Aqui, df_1 e df_2 são os conjuntos das variações não positivas de f_1 e f_2 respectivamente.

Após a determinação de df_1 e df_2, verifica-se se $\min\limits_{v \in M(v_k)} \min\limits_{j=1,2} df_j(v)$ é zero, ou seja, verifica-se a situação em que, para todo $v_k \in M(v_k)$, não é possível continuar melhorando nenhum objetivo, fazendo com que o procedimento termine. No caso em que o valor mínimo é diferente de zero, tem-se a situação em que é possível melhorar um objetivo sem piorar o outro objetivo.

Se $\min\limits_{v \in M(v_k)} df_1(v)$ for zero, então f_2 pode ser melhorado e f_1 é inalterável. Se $\min\limits_{v \in M(v_k)} df_2(v)$ for zero, então f_1 pode ser melhorado e f_2 é inalterável. Senão, é possível melhorar qualquer objetivo sem piorar o outro (existem duas direções de busca). Nesse caso, deve-se optar por seguir uma das direções com a qual se obtém melhores resultados; em nosso caso optamos por melhorar f_1.

5.2.2.2 Melhorar alocação de máquinas tri-objetivo

O esquema geral do procedimento para melhorar alocação de máquinas para os problemas fJSP, iRS/OS e APS, quando se considera simultaneamente três medidas de desempenho, é apresentado a seguir.

Procedimento: Newton adaptado para o fJSP, iRS/OS e APS tri-objetivo

Input: s^*, v^*;

Output: v^*;

Início

$v_0 \leftarrow v^*, k \leftarrow 0$, descida ← verdadeiro;

enquanto (descida = verdadeiro) **faça**

$df_1 \leftarrow \emptyset, df_2 \leftarrow \emptyset, df_3 \leftarrow \emptyset$;

$\theta \leftarrow \min_{v \in M(v_k)} \max_{j=1,2,3} (f_j(s^*, v) - f_j(s^*, v_k))$;

se $(\theta < 0)$ **então**

$v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin} \max_{j=1,2,3} (f_j(s^*, v) - f_j(s^*, v_k))$;

senão se $(\theta = 0)$ **então**

para cada $v \in M_{(v_k)}$ **faça**

se ($\max_{j=1,2,3} (f_j(s^*, v) - f_j(s^*, v_k)) = 0$) **então**

$df_1 \leftarrow df_1 \cup \{f_1(s^*, v) - f_1(s^*, v_k)\}$;

$df_2 \leftarrow df_2 \cup \{f_2(s^*, v) - f_2(s^*, v_k)\}$;

$df_3 \leftarrow df_3 \cup \{f_3(s^*, v) - f_3(s^*, v_k)\}$;

senão

$df_1 \leftarrow df_1 \cup \{0\}, df_2 \leftarrow df_2 \cup \{0\}, df_3 \leftarrow df_3 \cup \{0\}$;

fim se;

fim para cada;

se ($\min_{v \in M(v_k)} \min_{j=1,2,3} (df_j(v)) = 0$) **então** descida ← falso;

senão se ($\min_{v \in M(v_k)} \min_{j=1,2} (df_j(v)) = 0$) **então** $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin}\, df_3(v)$;

senão se ($\min_{v \in M(v_k)} \min_{j=1,3} (df_j(v)) = 0$) **então** $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin}\, df_2(v)$;

senão se ($\min_{v \in M(v_k)} \min_{j=2,3} (df_j(v)) = 0$) **então** $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin}\, df_1(v)$;

senão se ($\min_{v \in M(v_k)} df_1(v) = 0$) **então**

$\theta_{23} \leftarrow \min_{v \in M(v_k)} \max_{j=2,3} (df_j(v))$;

se $(\theta_{23} < 0)$ **então** $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin} \max_{j=2,3} (df_j(v))$;

senão $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin}\, df_2(v)$;

fim se;

senão se ($\min\limits_{v \in M(v_k)} df_2(v) = 0$) **então**

$\theta_{13} \leftarrow \min\limits_{v \in M(v_k)} \max\limits_{j=1,3}(df_j(v))$;

se $(\theta_{13} < 0)$ **então** $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin} \max\limits_{j=1,3}(df_j(v))$;

senão $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin}\ df_1(v)$;

fim se;

senão se ($\min\limits_{v \in M(v_k)} df_3(v) = 0$) **então**

$\theta_{12} \leftarrow \min\limits_{v \in M(v_k)} \max\limits_{j=1,2}(df_j(v))$;

se $(\theta_{12} < 0)$ **então** $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin} \max\limits_{j=1,2}(df_j(v))$;

senão $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin}\ df_1(v)$;

fim se;

senão $v_{k+1} \leftarrow \underset{v \in M(v_k)}{argmin}\, df_1(v)$;

fim se;

senão descida ← falso;

fim se;

se (descida = verdadeiro) **então** $k \leftarrow k + 1$;

fim se;

fim enquanto;

$v^* \leftarrow v_k$;

Fim

No procedimento, v denota cada vizinho de $v_k \in M(v_k)$ na iteração k; f_1, f_2 e f_3 são as funções objetivo a minimizar; θ mede a variação de f_1, f_2 e f_3 entre a solução corrente (s^*, v_k) e as soluções vizinhas (s^*, v). Se θ for negativo, significa que é possível reduzir simultaneamente f_1, f_2 e f_3. Se θ for positivo, então, pelo menos uma função objetivo piora. Se θ for zero, então, é possível melhorar no máximo dois objetivos ou simplesmente nenhum objetivo melhora.

Aqui, θ_{23} mede a variação de f_2 e f_3 entre a solução corrente (s^*, v_k) e as soluções vizinhas (s^*, v), quando f_1 não pode melhorar; θ_{13} mede a variação de

f_1 e f_3 entre a solução corrente (s^*, v_k) e as soluções vizinhas (s^*, v), quando f_2 não pode melhorar e θ_{12} mede a variação de f_1 e f_2 entre a solução corrente (s^*, v_k) e as soluções vizinhas (s^*, v), quando f_3 não pode melhorar.

Aqui, df_1, df_2 e df_3 são os conjuntos das variações não positivas de f_1, f_2 e f_3 respectivamente.

Após a determinação de df_1, df_2 e df_3, verifica-se se $\min\limits_{v \in M(v_k)} \min\limits_{j=1,2,3} df_j(v)$ é zero, ou seja, verifica-se a situação em que, para todo $v_k \in M(v_k)$, não é possível continuar melhorando nenhum objetivo, fazendo com que o procedimento termine. No caso em que o valor mínimo é diferente de zero, tem-se a situação em que é possível melhorar um ou dois objetivos sem piora os demais objetivos.

Se $\min\limits_{v \in M(v_k)} \min\limits_{j=1,2} df_j(v)$ for zero, então f_3 pode ser melhorado, e f_1 e f_2 são inalteráveis. Se $\min\limits_{v \in M(v_k)} \min\limits_{j=1,3} df_j(v)$ for zero, então f_2 pode ser melhorado, e f_1 e f_3 são inalteráveis. Se $\min\limits_{v \in M(v_k)} \min\limits_{j=2,3} df_j(v)$ for zero, então f_1 pode ser melhorado, e f_2 e f_3 são inalteráveis.

Se $\min\limits_{v \in M(v_k)} df_1(v)$ for zero, então f_2 e f_3 podem ser melhorados e f_1 é inalterável. Agora, deve-se calcular θ_{23}. Se θ_{23} for negativo, então f_2 e f_3 podem ser melhorados simultaneamente. Se θ_{23} for zero, então f_2 pode ser melhorado mantendo f_3 inalterado ou f_3 pode ser melhorado mantendo f_2 inalterado. Nesse caso, deve-se optar por seguir uma das direções com a qual se obtém melhores resultados; em nosso caso optamos por melhorar f_2.

Se $\min\limits_{v \in M(v_k)} df_2(v)$ for zero, então f_1 e f_3 podem ser melhorados e f_2 é inalterável. Agora, deve-se calcular θ_{13}. Se θ_{13} for negativo, então f_1 e f_3 podem ser melhorados simultaneamente. Se θ_{13} for zero, então f_1 pode ser melhorado mantendo f_3 inalterado ou f_3 pode ser melhorado mantendo f_1 inalterado. Nesse caso, deve-se optar por seguir uma das direções com a qual se obtém melhores resultados, em nosso caso optamos por melhorar f_1.

Se $\min\limits_{v \in M(v_k)} df_3(v)$ for zero, então f_1 e f_2 podem ser melhorados e f_3 é inalterável. Agora, deve-se calcular θ_{12}. Se θ_{12} for negativo, então f_1 e f_2 podem ser melhorados simultaneamente. Se θ_{12} for zero, então f_1 pode ser melhorado mantendo f_2 inalterado ou f_2 pode ser melhorado mantendo f_1 inalterado. Nesse caso, deve-se optar por seguir uma das direções com a qual se obtém melhores resultados; em nosso caso optamos melhorar f_1.

A última situação assegura que é possível melhorar qualquer objetivo e que é possível melhorar no máximo um dos demais objetivos. Nesse caso, deve-se optar por seguir uma das direções com a qual se obtém melhores resultados; em nosso caso optamos por melhorar f_1.

5.2.3
Melhorar seqüência de operações – Procedimento proposto auxiliar

O procedimento proposto resolve um problema auxiliar bi-objetivo, onde o primeiro objetivo f_1 é equivalente ao objetivo original do problema de minimização e o segundo objetivo f_4 é um objetivo auxiliar não conflitante com f_1. A idéia do procedimento proposto é minimizar primeiramente f_1 e depois minimizar f_4 sem piorar f_1.

Considerando v^* fixa, o procedimento parte de uma seqüência inicial de operações $s_0 = s^*$, para $k = 0$. Em cada iteração k gera-se a vizinhança $N(s_k)$ da seqüência s_k e testando o parâmetro θ escolhe-se o melhor vizinho de $N(s_k)$ de modo que minimize f_1 (neste caso f_4 pode piorar), na situação que não seja possível melhorar f_1, escolhe-se o melhor vizinho de $N(s_k)$ que minimize f_4 sem piorar f_1. O melhor vizinho de $N(s_k)$ será s_{k+1} e k passa a ser $k + 1$. O procedimento continuará enquanto for possível reduzir f_1 ou reduzir f_4 sem piorar f_1 (caso em que descida=verdadeiro). Finalmente, ao término do procedimento a melhor seqüência s_k será atribuída a s^*. A vizinhança $N(s_k)$ é gerada pelo mesmo método descrito nas seções 2.2.5 – 2.3.5 – 2.4.5 para o fJSP, iRS/OS e APS respectivamente. O esquema geral do procedimento é descrito a seguir.

Procedimento: Melhorar seqüência de operações para o fJSP, iRS/OS e APS

Input: s^*, v^*;

Output: s^*;

Início

$s_0 \leftarrow s^*, k \leftarrow 0$, descida ← verdadeiro;

enquanto (descida = verdadeiro) **faça**

$df_4 \leftarrow \emptyset$;

$\theta \leftarrow \min_{s \in N(s_k)} (f_1(s, v^*) - f_1(s_k, v^*))$;

se $(\theta < 0)$ **então**

$s_{k+1} \leftarrow \underset{s \in N(s_k)}{argmin}(f_1(s, v^*) - f_1(s_k, v^*))$;

senão se $(\theta = 0)$ **então**

para cada $s \in N_{(s_k)}$ **faça**

se ($\max_{j=1,4}(f_j(s, v^*) - f_j(s_k, v^*)) = 0$) **então**

$df_4 \leftarrow df_4 \cup \{f_4(s, v^*) - f_4(s_k, v^*)\}$;

senão

$df_4 \leftarrow df_4 \cup \{0\}$;

fim se;

fim para cada;

se ($\min_{s \in N(s_k)} df_4(s) = 0$) **então** descida ← falso;

senão $s_{k+1} \leftarrow \underset{s \in N(s_k)}{argmin}\, df_4(s)$;

fim se;

senão descida ← falso;

fim se;

se (descida = verdadeiro) **então** $k \leftarrow k + 1$;

fim se;

fim enquanto;

$s^* \leftarrow s_k$;

Fim

No procedimento, s denota cada vizinho de $s_k \in N(s_k)$ na iteração k; θ mede a variação de f_1 entre a solução corrente (s_k, v^*) e as soluções vizinhas

(s, v^*). Se θ for negativo, então, é possível reduzir f_1. Se θ for positivo, então f_1 piora. Se θ for zero, então, é possível melhorar f_4 sem piorar f_1 ou simplesmente nenhum objetivo melhora.

Aqui, df_4 é o conjunto das variações não positivas de f_4.

Após a determinação de df_4, verifica-se se $\min_{s \in N(s_k)} df_4(s)$ é zero, ou seja, verifica-se a situação em que para todo $s \in N(s_k)$, não é possível continuar melhorando f_4. No caso em que o valor mínimo é diferente de zero, tem-se a situação em que é possível melhorar f_4 sem piorar f_1.