2024 Winter Lab Seminar

Coupling Genetic Local Search and Recovering Beam Search Algorithms for Minimizing the Total Completion Time in the Single Machine Scheduling problem subject to Release Dates

Computers & Operations Research 39 (2012) 1257-1264

Rakrouki, M. A., Ladhari, T., & T'kindt, V.

January 11, 2024

Hyeon-Il Kim

Department of Industrial Engineering
Hanyang University
Seoul, Republic of Korea

PLI LAB
Production & Logistics Information

CONTENTS

1. Introduction

❖ Background

❑ Single machine scheduling problem

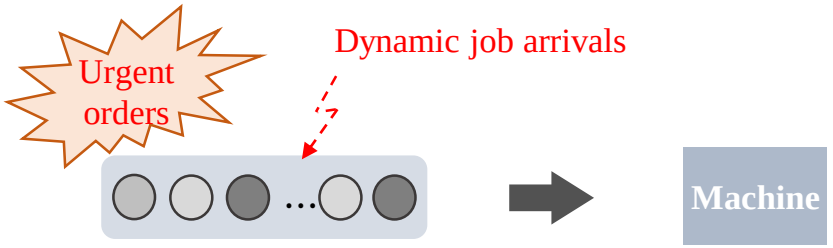
- Determine the sequence of n jobs to optimize a given performance measure
  - Pure sequencing problem (Ordering of the jobs)
    - ✓ Total number of distinct solutions = $n!$ (Permutation schedule)
- Assumptions for basic single machine scheduling problem
  - Zero ready times (Static) ↔ Nonzero ready times (Dynamic)
  - Sequence-independent setup times ↔ Sequence-dependent setup times
  - Deterministic job descriptors (processing times, due dates, etc.) ↔ Stochastic job descriptors
  - No preemption ↔ Preemption

n jobs (with a single operation)

2. System and Problem Descriptions

❖ System description

- ❑ Single machine scheduling problem with release dates ($1 \mid r_j \mid \sum C_j$)
  - Dynamic job arrivals: deterministic job release time
  - Complexity: NP-hard
  - Non-delay dispatching of SPT (Shortest Processing Time)
    - Heuristic (Optimal when release and processing times are agreeable)
      - ✓ $r_{[1]} \leq r_{[2]} \leq \cdots \leq r_{[n]}$ and $p_{[1]} \leq p_{[2]} \leq \cdots \leq p_{[n]}$



n jobs (with a single operation)

2. System and Problem Descriptions

❖ Problem description

- ❑ Decision variable
  - Start time of each job on single machine
- ❑ Objective function
  - Minimizing the total completion time
- ❑ Assumptions
  - All data deterministic and given in advance
  - Sequence-independent setup times
  - Negligible transportation times
  - Sufficient finite buffer capacity
  - No machine breakdowns

3. Solution Algorithms

❖ Constructive heuristics

❑ HC heuristic

- A necessary and sufficient condition for local optimality
- This condition is based on the priority function $PRTF(j,\Delta) = \max(\Delta, r_j) + p_j$

Theorem 1 (Chu [10]). Given j and k which have to be scheduled after time Δ, we have $C_{jk} \leq C_{kj}$ if and only if $PRTF(j,\Delta) \leq PRTF(k,\Delta)$. (local optimality condition)

Remark 1. The function $PRTF(j,\Delta)$ is a combination of the FIFO and SPT priority rules.

Remark 2. If $r_j = 0\ \forall j$, the necessary and sufficient condition becomes $p_j \leq p_i$. (SPT rule)

Remark 3. If $p_j = D\ (constant)\ \forall j$, the necessary and sufficient condition becomes $r_j(\Delta) \leq r_i(\Delta)$. (FIFO rule)

3. Solution Algorithms

❖ Constructive heuristics

❑ HC heuristic

Step 1. $\sigma = \emptyset,\ \bar{J} = J,\ \Delta = 0$.

Step 2. Arrange the jobs $j \in \bar{J}$ in non-decreasing order of $PRTF(j, \Delta)$.

Let $\Pi = (\pi(1), \pi(2), \cdots, \pi(n))$ denotes the resulting sequence.

Step 3. $\sigma = (\pi(1), \pi(2))$, and $\bar{J} = \bar{J} \backslash \{\pi(1), \pi(2)\}$.

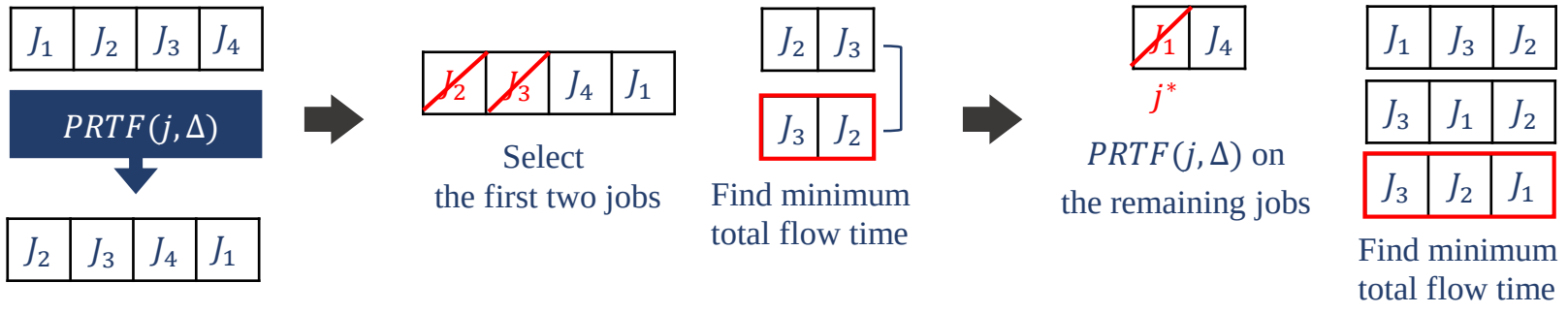
Step 4. $\Delta = C_\sigma$, Update $PRTF(j, \Delta)$ for all jobs $j \in \bar{J}$.

Step 5. Find the job j^* having the minimum $PRTF(j^*, \Delta)$ value. Set $\bar{J} = \bar{J} \backslash \{j^*\}$.

Step 6. Select the job j^* and insert it in $(|\sigma| + 1)$ possible positions of σ. (NEH procedure)

Step 7. If $\bar{J} = \emptyset$ **Then** stop **Else** go to **Step 4**.

- σ: partial sequence of the scheduled jobs
- $\bar{J}$: set of un scheduled jobs
- C_σ: completion time of σ



3. Solution Algorithms

❖ Constructive heuristics

❑ HCP heuristic

- Perturbation-based algorithm – instance perturbation
  - When a local optimum is reached, the instance data are marginally perturbed
  - The resulting new local optimum is than translated back into the original data
    - ✓ The HCP heuristic consists of randomly perturbing r_j and p_j as follows:

$$r_j' \in U[r_j, r_j + \frac{A \times r_j}{100}],\ p_j' \in U[p_j, p_j + \frac{B \times p_j}{100}]$$

    - ➢ Apply the HC heuristic on perturbed data and let S be the obtained sequence
    - ➢ Compute the total completion time on the original data

3. Solution Algorithms

❖ Constructive heuristics

❑ HCP heuristic

Step 1. Initialization

Construct a solution by means of the HC heuristic. Let C^* be the resulting total completion time

Step 2. Main loop

for $A \in \{3,4,5,6,7,8\}$ **do**

for $B \in \{3,4,5,6,7,8\}$ **do**

Generate the r_j' and p_j' values.

Apply the HC heuristic on perturbed data and let S be the obtained sequence.

Compute the total completion time $\sum C_j(S)$ on the original data.

Update $C^* = \min(C^*, \sum C_j(S))$.

end for

end for

3. Solution Algorithms

❖ Local search methods

- ❑ Embedding a local search algorithm within the HC heuristics (HCLS1 and HCLS2 algorithms)
  - ○ The goal is to improve the current solution by using a set of successive operations
    - Let σ be a partial solution and $V[\sigma]$ its neighborhood
      - ✓ $V[\sigma] = \{\pi / \exists i, j \in \sigma$ such that $\sigma = \sigma_1 i \sigma_2 j \sigma_3$ and $\pi = \sigma_1 j \sigma_2 i \sigma_3\}$
        - ➢ $\boldsymbol{Swap}(\sigma, i, j)$ operator: Swapping jobs in positions i and j in σ

Local Search Procedure(σ, k)

Input: Current solution σ and a number of iterations k.

while (k iterations have not been performed) **do**

Randomly select two positions i and j ($i \neq j$).

$\sigma' = \boldsymbol{Swap}(\sigma, i, j)$.

if $\sum C_j(\sigma') < C_j(\sigma)$ **then**

$\sigma = \sigma'$;

end if

end while

3. Solution Algorithms

❖ Local search methods

❑ HCLS1 algorithm

- The goal is to improve the current solution by using a set of successive operations

Step 1. $\sigma = \emptyset,\ \bar{J} = J,\ \Delta = 0$.

Step 2. Arrange the jobs $j \in \bar{J}$ in non-decreasing order of $PRTF(j, \Delta)$.

Let $\Pi = (\pi(1), \pi(2), \cdots, \pi(n))$ denotes the resulting sequence.

Step 3. $\sigma = (\pi(1), \pi(2))$, and $\bar{J} = \bar{J} \backslash \{\pi(1), \pi(2)\}$.

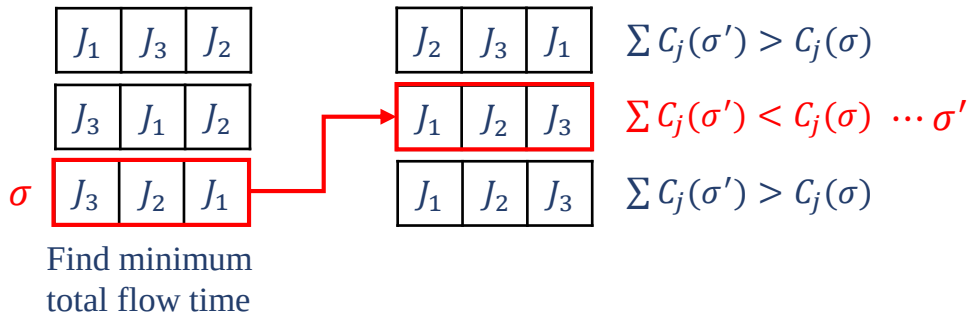
Step 4. $\Delta = C_\sigma$, Update $PRTF(j, \Delta)$ for all jobs $j \in \bar{J}$.

Step 5. Find the job j^* having the minimum $PRTF(j^*, \Delta)$ value. Set $\bar{J} = \bar{J} \backslash \{j^*\}$.

Step 6. Select the job j^* and insert it in $(|\sigma| + 1)$ possible positions of σ. (NEH procedure)

Step 7. Local Search Procedure $(\sigma, |\sigma|)$

Step 8. If $\bar{J} = \emptyset$ **Then** stop **Else** go to **Step 4**.



3. Solution Algorithms

❖ Local search methods

❑ HCLS2 algorithm

- The goal is to improve the current solution by using a set of successive operations

Step 1. $\sigma = \emptyset,\ \bar{J} = J,\ \Delta = 0$.

Step 2. Arrange the jobs $j \in \bar{J}$ in non-decreasing order of $PRTF(j, \Delta)$.

Let $\Pi = (\pi(1), \pi(2), \cdots, \pi(n))$ denotes the resulting sequence.

Step 3. $\sigma = (\pi(1), \pi(2))$, and $\bar{J} = \bar{J} \backslash \{\pi(1), \pi(2)\}$.

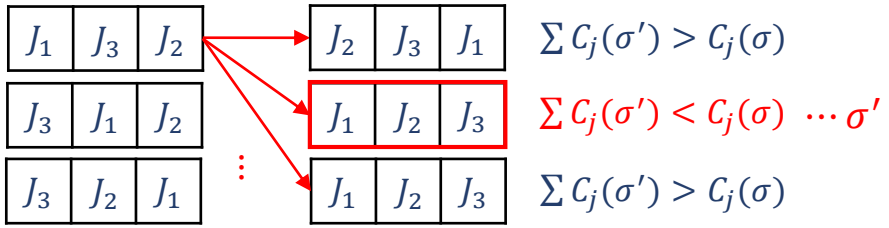
Step 4. $\Delta = C_\sigma$, Update $PRTF(j, \Delta)$ for all jobs $j \in \bar{J}$.

Step 5. Find the job j^* having the minimum $PRTF(j^*, \Delta)$ value. Set $\bar{J} = \bar{J} \backslash \{j^*\}$.

Step 6. Select the job j^* and insert it in $(|\sigma| + 1)$ possible positions of σ. (NEH procedure)

Step 6.1. Local Search Procedure $(\sigma, |\sigma|)$

Step 7. If $\bar{J} = \emptyset$ **Then** stop **Else** go to **Step 4**.

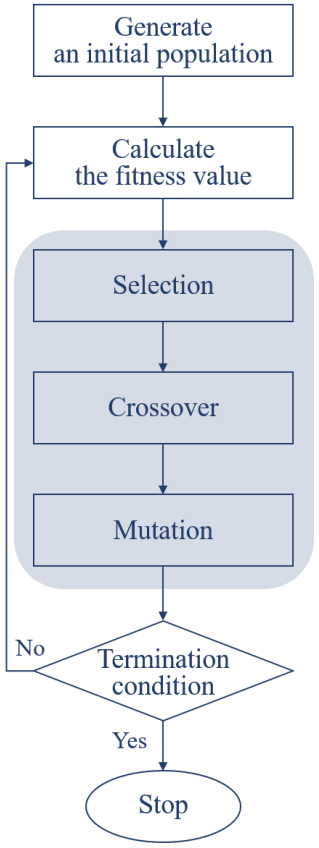


For each partial sequence σ with job j^* inserted in a given position, apply Local Search Procedure$(\sigma, |\sigma|)$

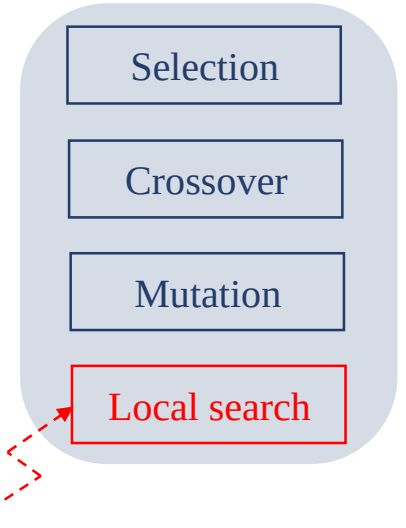
3. Solution Algorithms

❖ Genetic local search (GLS) algorithm

- ❑ A genetic algorithm and a local search in order to explore the solution space for good solutions



Improvement of the population



To improve generated offsprings

3. Solution Algorithms

❖ Genetic local search (GLS) algorithm

❑ Solution representation

- An example of solution representation with six jobs
  - A job sequence vector $\boldsymbol{J} = (j_{[1]}, j_{[2]}, \dots, j_{[n]})$, where $j_{[i]}$ denotes the index of the job in the ith position

J_4	J_6	J_1	J_5	J_2	J_3

❑ Generate an initial population

- An initial population consists in M solutions
  - $M-1$ chromosomes are generated randomly while a single chromosome is generated using the HC heuristic

J_4	J_6	J_1	J_5	J_2	J_3
⋮					
J_1	J_2	J_6	J_3	J_5	J_4

$M-1$ (Random generation)

J_5	J_2	J_3	J_4	J_1	J_6

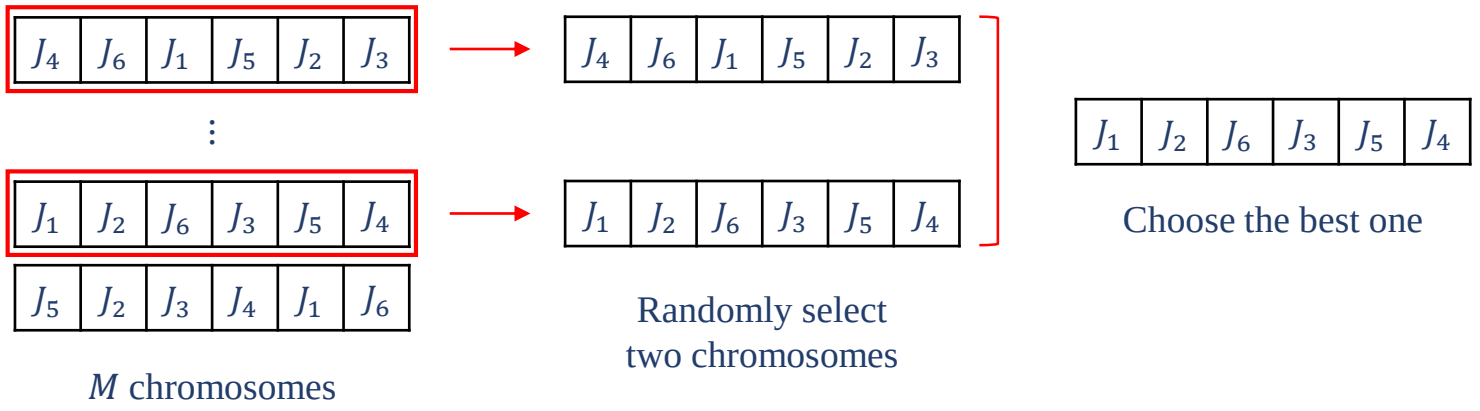
HC heuristic

3. Solution Algorithms

❖ Genetic local search (GLS) algorithm

❑ Selection

- ○ To generate a new population for the next generation
  - Binary tournament selection strategy, which selects an individual from a population based on their fitness value

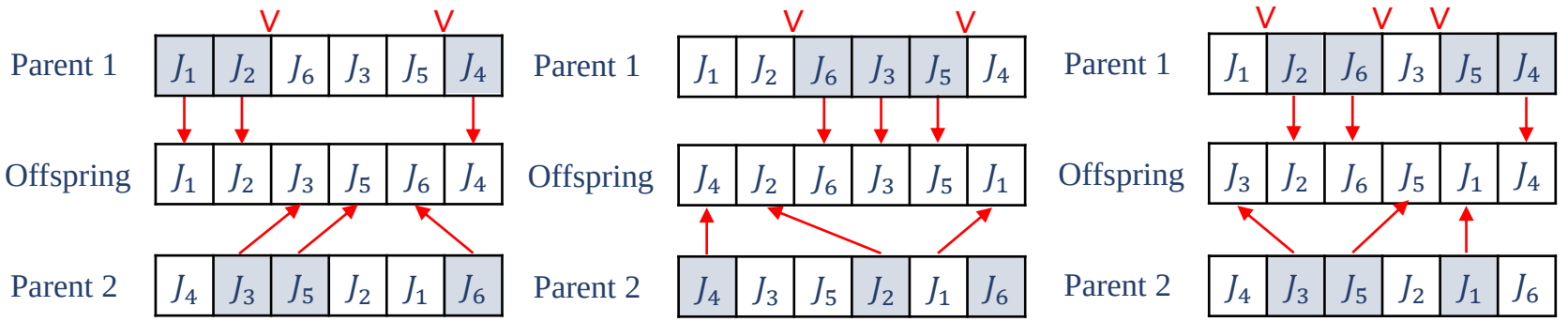


3. Solution Algorithms

❖ Genetic local search (GLS) algorithm

❑ Crossover

- Two good individuals are combined into the new individuals to preserve good information
  - Two-point crossover operators (C1 and C2) and Three-point crossover operator (C3)
  - Three crossover operators are available in the GLS algorithm but only one is selected at each time
    - ✓ The choice of one is done according to probabilities: $P_{C1} = P_{C2} = 0.3$ and $P_{C3} = 0.4$



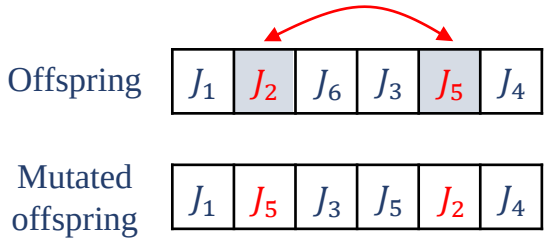
Two-point crossover operator C1 Two-point crossover operator C2 Three-point crossover operator C3

3. Solution Algorithms

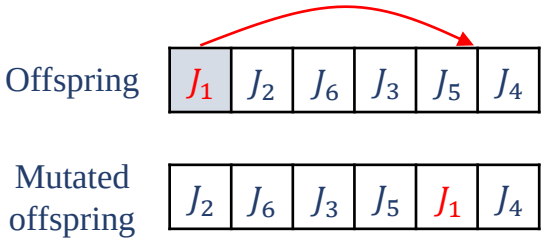
❖ Genetic local search (GLS) algorithm

❑ Mutation

- ○ The opportunity to escape from trapping the local optima and to increase diversity in the population
  - Exchange mutation operator (Mu1) and Insertion mutation operator (Mu2)
  - Two mutation operators are available in the GLS algorithm but only one is selected at each time
    - ✓ The choice of one is done according to probabilities: $P_{Mu1} = P_{Mu2} = 0.5$



Exchange mutation operator Mu1



Insertion mutation operator Mu2

3. Solution Algorithms

❖ Genetic local search (GLS) algorithm

❑ Local search procedure

- ○ The local search is done to enhance the intensification process in the genetic algorithm
  - • Two local search algorithm can be applied but only one is selected at each time
    - ✓ The choice of one is done according to probabilities: $P_{LS1} = P_{LS2} = 0.5$

Improvement_Procedure1(π)
Begin
Step 1: Generate k ($k = 30$) neighbors of a given solution π.
- $k/2$ of the neighbors are generated by **Exchanging**(π, i, j).
- $k/2$ of the neighbors are generated by **Swapping**(π, i, j).

Step 2: Insert the best neighbor in the population.
End

Improvement_Procedure2(n, π)
Begin
Step 1: $\pi' =$ **Exchange**(π, i, j).
- **If** Fitness_Function(π') < Fitness_Function(π) **then** replace π by π'($\pi \leftarrow \pi'$).

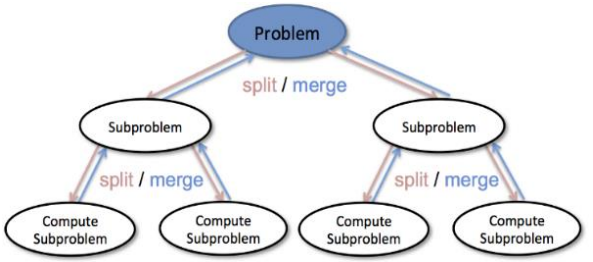
Step 3: **If** the termination condition (*n iterations where n is the size of* π) is not reached **then** goto **Step 1**.
Step 4: Insert π in the population.
End

3. Solution Algorithms

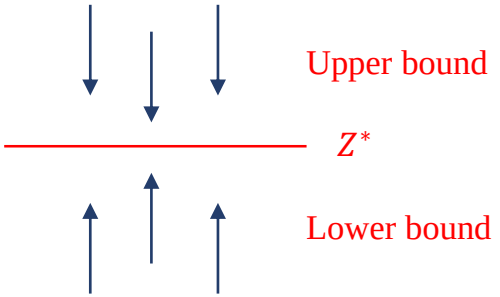
❖ Genetic recovering beam search algorithm

❑ Branch and bound (B&B) algorithm

- Enumeration technique that gives the optimal solutions for various combinatorial optimization problems
  - Divide and conquer
    - ✓ Break the problem into a series of smaller problems that are easier
    - ✓ Solve the smaller problems
    - ✓ Put the information together again to solve the original problem



  - Branching
    - ✓ Full enumeration of all possible solutions (Enumeration tree)
  - Bounding
    - ✓ Pruning by infeasibility
    - ✓ Pruning by optimality
    - ✓ Pruning by bound



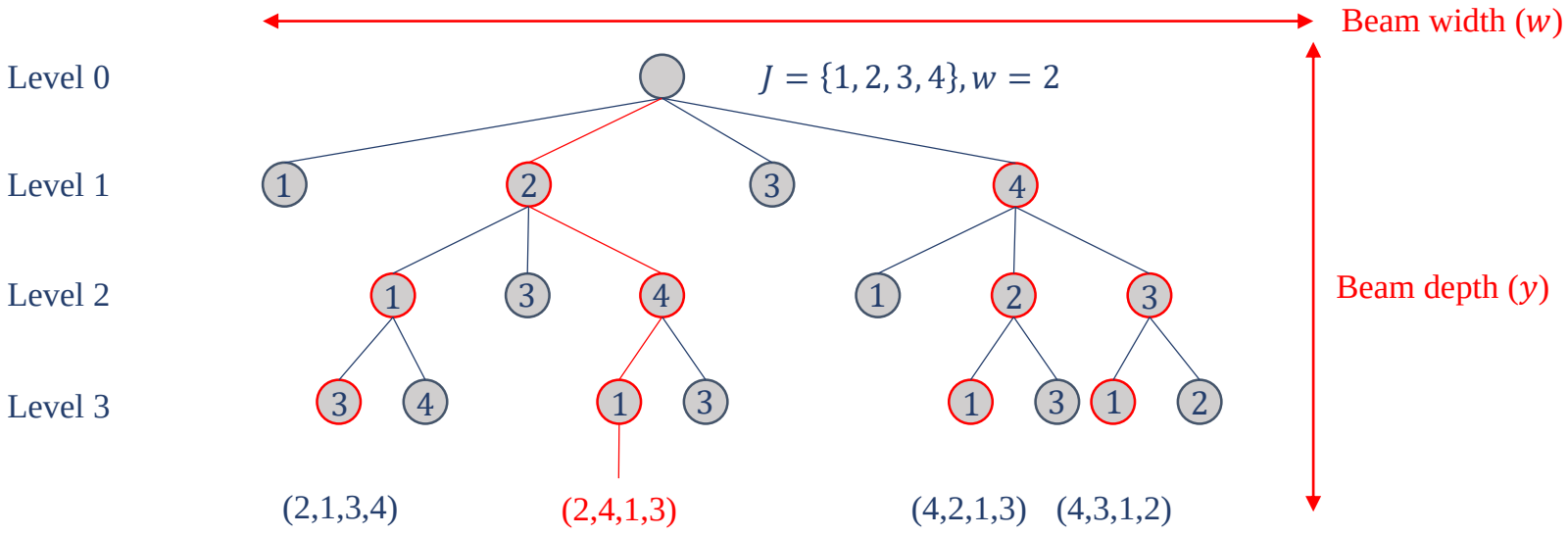
- ➢ Tight lower and upper bounds are a key for efficient B&B algorithms
- ➢ How to choose between a fast weak bound and a time-consuming stronger bound?

3. Solution Algorithms

❖ Genetic recovering beam search algorithm

❑ Beam search (BS) algorithm

- Truncated branch and bound algorithm
  - Powerfulness of the branching scheme of a B&B algorithm and heuristic evaluations to prune some branches
  - Only the most promising w nodes are retained for further branching at each level of the search tree (w: Beam width)
    - ✓ The remaining nodes are discarded and backtracking is not allowed
      - ➢ The performance and effectiveness of BS algorithm greatly depends on the node evaluation process



3. Solution Algorithms

❖ Genetic recovering beam search algorithm

❑ Beam search (BS) algorithm

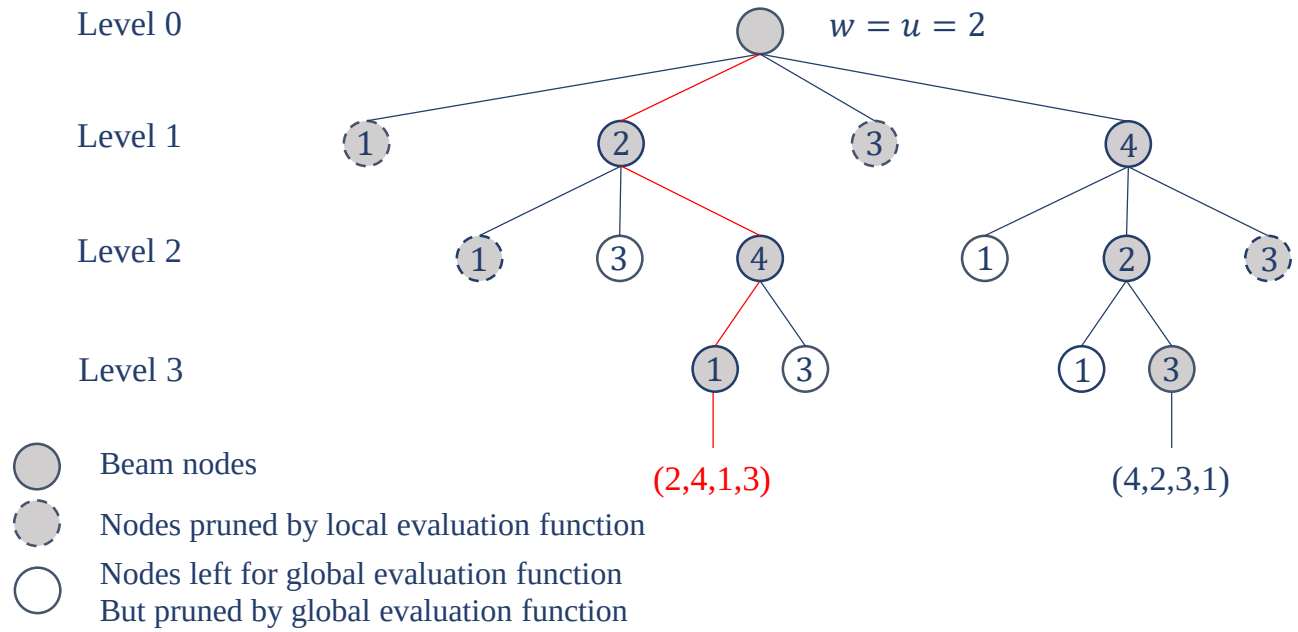
- ○ Truncated branch and bound algorithm
  - Two different types of node evaluation functions have been used in beam search algorithm
    - ✓ Priority evaluation function (= Local evaluation function)
      - ➢ Calculate an urgency rating for the job added to the current partial schedule
      - ➢ A local view because of only considering the next decision to be made (Dispatching heuristic)
    - ✓ Total cost evaluation function (= Global evaluation function)
      - ➢ Calculate an estimate of the minimum total cost of the best solution that can be reached from the current node
      - ➢ A global view because of considering from current partial schedule to a complete schedule (Primal bound)

3. Solution Algorithms

❖ Genetic recovering beam search algorithm

❑ Filtered Beam search (FBS) algorithm

- A filter is applied to heuristically prune some non-interesting nodes at each level of search tree
  - All nodes generated from a parent node are evaluated crudely by filtering procedure (local evaluation function)
  - Leaving filtered nodes u (filter width) for further accurate evaluation by global evaluation function



3. Solution Algorithms

❖ Genetic recovering beam search algorithm

❑ Recovering Beam search (RBS) algorithm

- BS and FBS algorithms cannot recover from wrong decisions
  - If a node leading to the optimal solution is fathomed, there is no way to reach that afterwards that solution
  - Main differences between the FBS and the RBS algorithms are:

1. Filtering procedure is no longer calculated by evaluation function but using a specific dominance properties
   - ✓ No value to assign to the filter width, only the nodes satisfying the dominance properties
2. Global evaluation uses both lower and upper bounds
   - ✓ Each node is evaluated by the function $V = (1 - \alpha) \cdot LB + \alpha \cdot UB \ (0 \leq \alpha \leq 1)$
3. Recovering procedure is used to recover the previous incorrect decisions
   - ✓ Local search is applied on the current partial solution

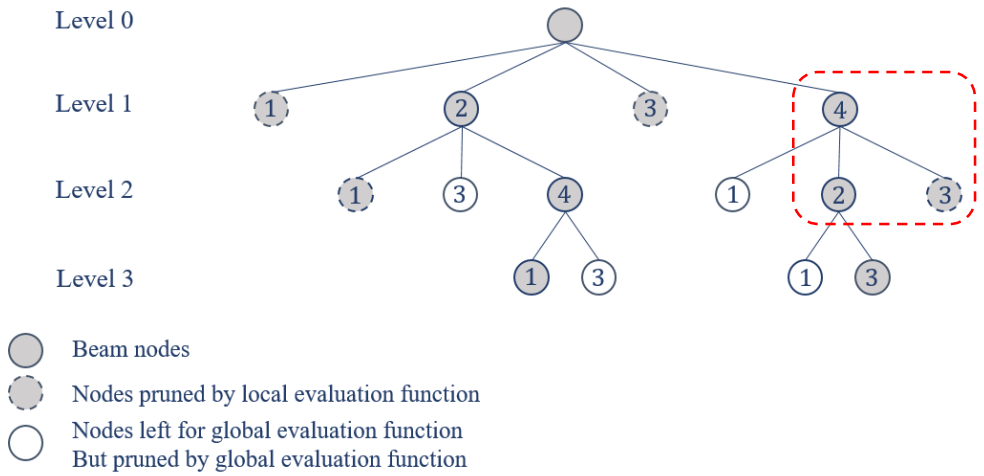
3. Solution Algorithms

❖ Genetic recovering beam search algorithm

❑ Recovering Beam search (RBS) algorithm

- BS and FBS algorithms cannot recover from wrong decisions
  - Filtering procedure: Specific dominance properties

Property 1. (Chu [11]). Given a branch σ and a pair of jobs $i, j \notin \sigma$, if $r_i(\sigma) \leq r_j(\sigma)$ and $r_i(\sigma) + p_i \leq r_j(\sigma) + p_j$ with at least one strict inequality, then the value of the SRPT lower bound for branch σi is never superior to the value of the SRPT lower bound for branch σj.



$\sigma = \{4\}$

$\sigma 2 = 11, \sigma 3 = 15$

$\sigma 3$ is dominated by $\sigma 2$

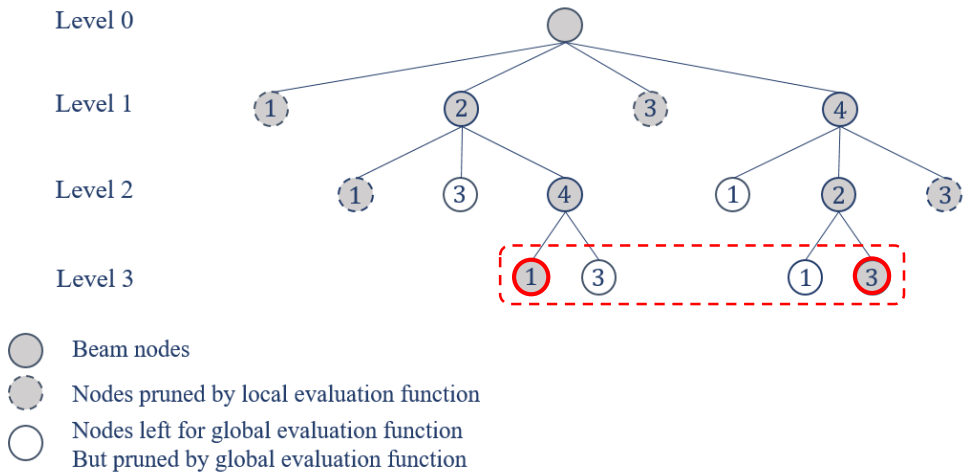
✓ The filter is based on a pseudo-dominance condition which means that nodes leading to optimal solutions can be discarded

3. Solution Algorithms

❖ Genetic recovering beam search algorithm

❑ Recovering Beam search (RBS) algorithm

- BS and FBS algorithms cannot recover from wrong decisions
  - Node evaluation: Both lower and upper bounds (Global evaluation)
    - ✓ Each node is evaluated by the function $V = (1-\alpha)\cdot LB + \alpha \cdot UB$ $(0 \le \alpha \le 1)$
      - ➢ UB: $PRTF(j, \Delta) = \max(\Delta, r_j) + p_j$
      - ➢ LB: SRPT (Shortest Remaining Processing Time) rule – Valid lower bound



Select the lowest value of function V

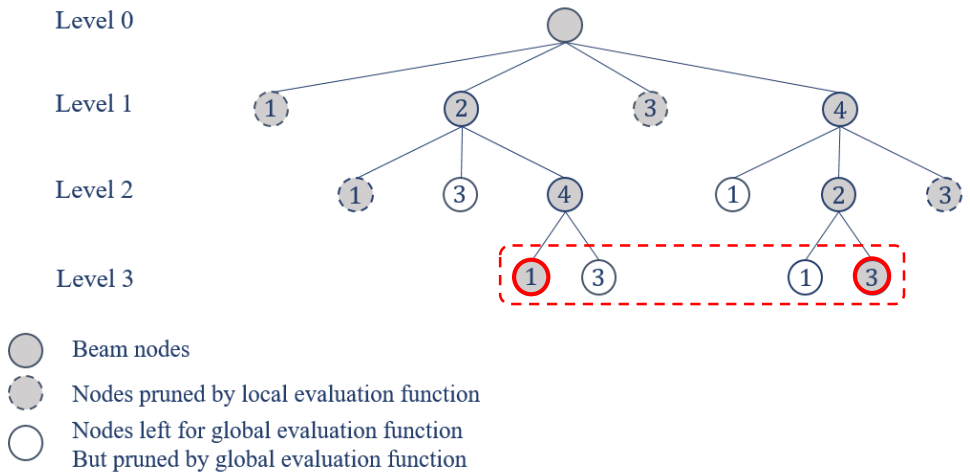
$Q = [\{2,4,1\}, \{4,2,3\}]$

3. Solution Algorithms

❖ Genetic recovering beam search algorithm

❑ Recovering Beam search (RBS) algorithm

- BS and FBS algorithms cannot recover from wrong decisions
  - Recovering procedure: Local search procedure
    - ✓ Each node is evaluated by the function $V = (1-\alpha)\cdot LB + \alpha \cdot UB$ $(0 \le \alpha \le 1)$
      - ➢ UB: $PRTF(j, \Delta) = \max(\Delta, r_j) + p_j$
      - ➢ LB: SRPT (Shortest Remaining Processing Time) rule – Valid lower bound



$Q = [\{2,4,1\}, \{4,2,3\}]$

For all $\sigma \in Q$, apply a local search on σ and let σ' be the obtained partial solution that dominates σ having the same level of the search tree

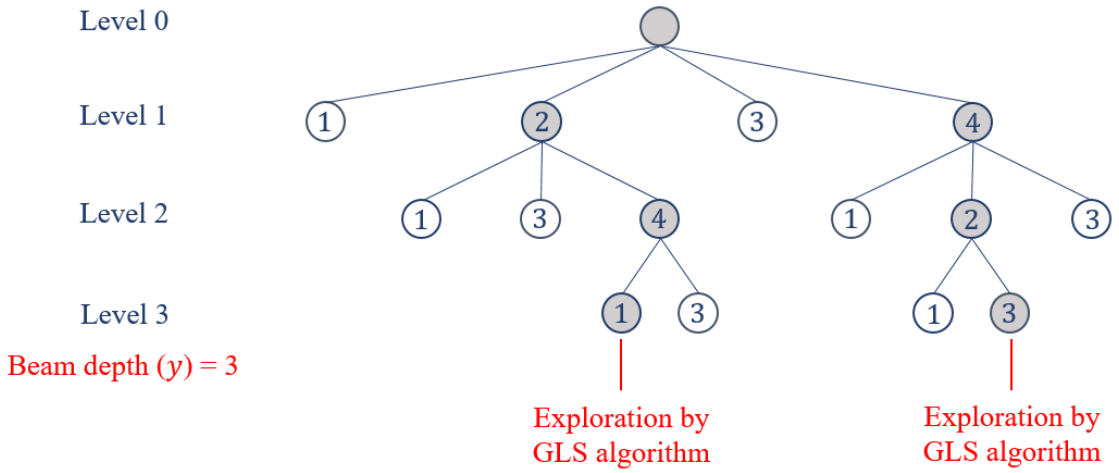
$\sigma = \{2,4,1\} \rightarrow \sigma' = \{2,4,3\}$
$\sigma = \{4,2,3\} \rightarrow \sigma' = \{4,2,1\}$

3. Solution Algorithms

❖ Genetic recovering beam search algorithm

❑ Genetic Recovering Beam search (GRBS) algorithm

- High capability of the RBS algorithm and the effectiveness of the GLS algorithm to compute good local optimum
  - RBS algorithm is ran up to nodes of a given level y (beam depth) are obtained
  - On all the w retained nodes at this level, the GLS algorithm is applied to heuristically explore the sub-problems
    - ✓ Filtering procedure can fathom the nodes leading to an optimal solution
    - ✓ Filtering procedure also fathom the nodes that may lead the GLS algorithm to compute good solutions
      - ➢ Filtering procedure was not considered in this study



4. Experiment Results

❖ Design of experiments

❑ Environments

- PC with Intel Pentium IV at 3.0 GHz 1GB RAM
- Programmed by C

❑ Test data

- Number of jobs (n): 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150 and 200
- Number of each instances: 20
- Processing time (p_i) ~ $DU(1, 100)$
- Release time (r_i) ~ $DU(0, 50.5nR)$
  where R = {0.2, 0.4, 0.6, 0.8, 1.0, 1.25, 1.5, 1.75, 2.0 and 3.0}

❑ Performance measures

- Average percentage gaps
  - $\delta = \frac{UB-OPT}{OPT} \times 100$

  UB: total completion time obtained from heuristic
  OPT: Optimal total completion time obtained from B&B algorithm of T'kindt et al. [37]
- Average CPU time

4. Experiment Results

❖ Preliminary experiments

❑ Parameter setting

- RBS parameters
  - Beam width (w) = Beam depth (y) = $n/4$
  - $\alpha = 0.5$
- GLS parameters (Rakrouki and Ladhari [33])
  - Maximum number of generations = $n \times 10$
  - Maximum of consecutive generations without improving the best solution of the population = $n \times 5$
  - Population size (M) = 200
  - Crossover probability (P_c) = 0.9
    - ✓ Two-point crossover probability (P_{c1}) = 0.3
    - ✓ Two-point crossover probability (P_{c2}) = 0.3
    - ✓ Three-point crossover probability (P_{c3}) = 0.4
  - Mutation probability (P_{Mu}) = 0.7
    - ✓ Exchange mutation probability (P_{Mu1}) = 0.5
    - ✓ Insertion mutation probability (P_{Mu2}) = 0.5

4. Experiment Results

❖ Performance of the constructive heuristics

- ❑ Comparisons among HC, HCP and GL heuristics
  - HCP heuristic dominates the HC heuristic both in terms of the optimality gap and the percentage of optimal solution
  - GL heuristic outperforms the HC heuristic in that it finds more often optimal solutions
  - HC and HCP heuristics outperform GL heuristic

n	*GL(MB)*			*HC*			*HCP*		
	gapOpt	*OptS*	*Time*	*gapOpt*	*OptS*	*Time*	*gapOpt*	*OptS*	*Time*
10	1.3059	82.5	0	0.4914	66.5	0	0.2890	73	0.005
20	1.8566	63.5	0	0.5498	45.5	0	0.2965	55.5	0.021
30	1.9357	41.5	0	0.5609	25.5	0	0.3416	35	0.057
40	1.3544	32.5	0	0.5991	24	0	0.3518	31	0.118
50	0.8271	27.5	0	0.5271	14.5	0	0.2630	18	0.204
60	0.9920	14	0	0.5379	11	0	0.2663	16	0.326
70	0.8837	15	0	0.4892	14.5	0	0.1825	15	0.483
80	0.6612	3	0	0.4291	6	0	0.2332	7	0.689
90	0.4898	5.5	0	0.4314	7	0	0.2010	9	0.943
100	0.5643	5.5	0	0.3918	5.5	0	0.2133	7.5	1.276
Mean	**1.0871**	**29.1**	**0**	**0.5008**	**22.0**	**0**	**0.2638**	**26.7**	**0.412**
150	0.1227	–	0.071	0.4004	–	0.020	0.2600	–	3.780
200	0.1051	–	0.169	0.3282	–	0.046	0.2044	–	8.316
Mean	**0.1139**	–	**0.120**	**0.3643**	–	**0.033**	**0.2322**	–	**6.048**

SRPT lower bound (LB)

4. Experiment Results

❖ Performance of the local search heuristics

- ❑ Comparisons among LA, HCLS1 and HCLS2 heuristics
  - HCLS2 outperforms HCLS1 both in terms of the average optimality gap and the percentage of optimal solutions
  - The average CPU time is drastically increases
  - HCLS1 seems to be a better candidate for being a good heuristic to solve the large size instances

n	*LA(MB)*			*HCLS1*			*HCLS2*		
	gapOpt	*OptS*	*Time*	*gapOpt*	*OptS*	*Time*	*gapOpt*	*OptS*	*Time*
10	0.0520	98.5	0.001	0.1654	87.5	0.000	0.0123	94.5	0.000
20	0.1723	91	0.006	0.0834	86.5	0.001	0.0069	95	0.006
30	0.2108	84	0.033	0.2129	60	0.003	0.0167	83	0.038
40	0.1684	77.5	0.104	0.1471	53	0.008	0.0155	74.5	0.147
50	0.0958	72	0.29	0.1731	37.5	0.019	0.0209	64	0.433
60	0.1344	62.5	0.643	0.1650	34	0.037	0.0153	60	1.075
70	0.1305	61	1.259	0.1394	35.5	0.066	0.0162	54	2.302
80	0.1279	46	2.313	0.1281	28	0.113	0.0139	51	4.393
90	0.1143	49	4.181	0.1411	25	0.179	0.0121	44.5	7.814
100	0.1213	43	5.39	0.1283	29.5	0.272	0.0120	46	13.568
Mean	**0.1328**	**68.5**	**1.422**	**0.1484**	**47.7**	**0.070**	**0.0142**	**66.7**	**2.977**
150	0.1028	–	27.979	0.1815	–	0.977	0.1048	–	105.224
200	0.0807	–	106.662	0.1670	–	3.127	0.0833	–	441.607
Mean	**0.0917**	–	**67.320**	**0.1742**	–	**2.052**	**0.0940**	–	**273.416**

4. Experiment Results

❖ Performance of the GLS and GRBS heuristics

❑ Comparisons among meta-heuristics and GRBS algorithm

- RBS outperforms the tabu search [23] in terms of the average deviation to the optimal solution
- GLS outperforms the RBS both in terms of the average optimality gap and the percentage of optimal solutions
- The best is GRBS which outperforms GLS in terms of solution quality

n	*TS*			*RBS*			*GLS*			*GRBS*		
	gapOpt	*OptS*	*Time*	*gapOpt*	*OptS*	*Time*	*gapOpt*	*OptS*	*Time*	*gapOpt*	*OptS*	*Time*
10	0.0000	100	0.000	0.0122	96	0.000	0.0000	100	0.242	0.0000	100	0.119
20	0.0000	100	0.000	0.0116	91.5	0.000	0.0000	100	0.942	0.0000	100	0.657
30	0.0000	100	0.003	0.0246	71	0.005	0.0000	100	2.033	0.0000	100	2.517
40	0.0020	99	0.207	0.0203	66	0.005	0.0012	97.5	3.962	0.0003	98.5	8.239
50	0.0040	98	0.834	0.0189	60.5	0.015	0.0014	93.5	4.802	0.0005	95	17.846
60	0.0120	98	2.320	0.0171	57.5	0.020	0.0012	92.5	7.667	0.0016	92.5	40.052
70	0.0220	94	4.510	0.0187	47	0.035	0.0022	90.5	12.782	0.0010	93	81.341
80	0.1340	80	7.145	0.0196	36.5	0.060	0.0024	88.5	19.696	0.0015	87	147.191
90	0.3280	70	9.659	0.0135	42.5	0.090	0.0031	83	29.775	0.0013	84	242.648
100	0.9910	60	12.597	0.0141	38.5	0.085	0.0029	76.5	43.394	0.0025	77	427.722
Mean	***0.1493***	***89.9***	***3.727***	***0.0171***	***60.7***	***0.032***	***0.0014***	***92.2***	***12.530***	***0.0009***	***92.7***	***96.833***
150	0.1074	–	17.226	0.1028	–	35.969	0.0973	–	186.770	0.0573	–	1791.762
200	0.0909	–	64.894	0.0807	–	137.124	0.0763	–	652.160	0.0663	–	4035.012
Mean	***0.0992***	–	***41.060***	***0.0917***	–	***86.547***	***0.0868***	–	***419.465***	***0.0618***	–	***2913.387***

5. Conclusions

❖ Summary

- ❑ Dynamic single machine scheduling problem
- ❑ Solution approaches
  - o New priority rule based constructive heuristics: HC and HCP heuristics
  - o Two local search procedures: HCLS1 and HCLS2 heuristics
  - o Hybrid meta-heuristics: Genetic algorithm and local search procedures (GLS algorithm)
  - o Hybrid solution algorithm: Recovering beam search and GLS algorithms (GRBS algorithm)
- ❑ GRBS algorithm consistently yield optimal or near-optimal solutions and outperformed the best-know ones

❖ Further researches

- ❑ More efficient GRBS algorithm in order to keep the same solution quality and to decrease the CPU time

❖ Advantages & Disadvantages

- ❑ GLS algorithm is not seen as an effective procedure for RBS algorithm
- ❑ Parameter values for the algorithms

Thank You!

Q&A


HANYANG UNIVERSITY
한양
1939


PLI LAB
Production & Logistics Information