applied sciences

MDPI

Article

GRASP and Iterated Local Search-Based Cellular Processing algorithm for Precedence-Constraint Task List Scheduling on Heterogeneous Systems

Alejandro Santiago [1,*], J. David Terán-Villanueva [2], Salvador Ibarra Martínez [2], José Antonio Castán Rocha [2], Julio Laria Menchaca [2], Mayra Guadalupe Treviño Berrones [2] and Mirna Ponce-Flores [3]

1 Information Technology Engineering, Polytechnic University of Altamira, Altamira 89602, Mexico
2 Facultad de Ingeniería, Universidad Autónoma de Tamaulipas, Tampico 89339, Mexico; jdteran@docentes.uat.edu.mx (J.D.T.-V.); sibarram@docentes.uat.edu.mx (S.I.M.); jacastan@docentes.uat.edu.mx (J.A.C.R.); jlaria@docentes.uat.edu.mx (J.L.M.); mgtrevino@docentes.uat.edu.mx (M.G.T.B.)
3 Graduate Program Division, Tecnológico Nacional de México/Instituto Tecnológico de Ciudad Madero, Cd. Madero 89440, Mexico; D94070293@cdmadero.tecnm.mx
* Correspondence: aurelio.santiago@upalt.edu.mx

Received: 28 September 2020; Accepted: 19 October 2020; Published: 25 October 2020



Abstract: High-Performance Computing systems rely on the software's capability to be highly parallelized in individual computing tasks. However, even with a high parallelization level, poor scheduling can lead to long runtimes; this scheduling is in itself an NP-hard problem. Therefore, it is our interest to use a heuristic approach, particularly Cellular Processing Algorithms (CPA), which is a novel metaheuristic framework for optimization. This framework has its foundation in exploring the search space by multiple Processing Cells that communicate to exploit the search and in the individual stagnation detection mechanism in the Processing Cells. In this paper, we proposed using a Greedy Randomized Adaptive Search Procedure (GRASP) to look for promising task execution orders; later, a CPA formed with Iterated Local Search (ILS) Processing Cells is used for the optimization. We assess our approach with a high-performance ILS state-of-the-art approach. Experimental results show that the CPA outperforms the previous ILS in real applications and synthetic instances.

Keywords: cellular processing algorithms; processing cell; iterated local search; GRASP; scheduling; makespan; HPC (High-Performance Computing); DAG (Directed Acyclic Graph)

1. Introduction

According to the website www.top500.org, the supercomputer Fugaku from the Fujitsu RIKEN Center for Computational Science in Japan consists of 7,299,072 processing units. The above gives an outstanding parallel computing power. However, without accurate and efficient scheduling methods, parallel programs can be very computationally inefficient. In this paper, we approach the precedence-constraint task scheduling of parallel programs on systems formed by heterogeneous processing units to minimize the final computing time [1]. Scheduling is a well-known NP-hard optimization problem [2], as well as the task scheduling for parallel systems [3]. Therefore, different scheduling approaches were developed for this problem: Heuristics [4–7], Local Searches [8–10], and Metaheuristics [11–15]. Unfortunately, the wide variety of works in the state-of-the-art differs in their objective definitions and uses different sets of instances. From the relevant works in the state-of-the-art, we highlight [11]. To our knowledge, it is the only work that compares the obtained

results against the optimal values for their synthetic instances, which are fourteen scheduling problem instances that are easily reproducible. In [11], four high-performance Iterated Local Search (ILS) algorithms are proposed, the best ILS nearly reaches all the optimal values, obtaining an approximation factor of 1.018 (desirable values are near to 1.0). Thus, we assess our proposal with the best-proposed algorithm in [11].

Scheduling precedence-constraint tasks for heterogeneous systems has been addressed with a lot of variants: energy-aware [16], idle energy-aware [8,17], energy constraint [18], communication delays and energy [19], budget constraint [20], fault-tolerant [21], security-aware [22], volunteer computing systems [23], among others. A related problem is the job shop problem [24]. However, in a parallel program, scheduling is a single job (the parallel program) without dedicated machines to a single task. On the other hand, in scheduling, any machine can compute any task, similar to the job shop's flexible problem [25] but with a single job. In Section 2, we detail our studied version of scheduling precedence-constraint tasks on heterogeneous systems.

In this paper, we use a recent metaheuristic optimization framework called Cellular Processing Algorithm (CPA) [26,27]. CPAs are more like a framework than a strict algorithm. The main idea is to cycle between exploring the search space with multiple limited-effort algorithms (Processing Cells) and sharing information (communication) among these Processing Cells. The limited-effort algorithms can explore the search space independently from one another and communicate their findings using shared memory or other combining methods. However, despite the communication, the Processing Cells should keep looking independently and not as a whole algorithm. In this way, we avoid the computational cost of converging the whole population (when the solutions reach the same search space area) or exploring the search space with a single solution algorithm (like Greedy Randomized Adaptive Search Procedure (GRASP) or Iterated Local Search).

As stated before, CPAs are flexible, e.g., a cellular processing algorithm can initialize each Processing Cell with different heuristics to ensure the splitting of the search space among the Processing Cells [28]. Additionally, the Processing Cells can be homogeneous or heterogeneous, which means that the Processing Cells can be multiple instances of the same algorithm [29] or completely different algorithms [30]. Furthermore, we can create a soft-heterogeneous CPA with differently-configured instances of the same algorithm. Thus, with all its flexibility, there is still much to research for CPAs and its applications.

The remainder of the paper is organized as follows. Section 2 details the studied precedence-constraint task scheduling problem on heterogeneous systems. In Section 3, we introduce the related and proposed algorithms for our experimentation. Section 4 contains the experimental setup, and parameter settings. Section 5 analyzes the experimental results according to the achieved median values using non-parametric statistical tests. Finally, Section 6 gives our conclusions and future work on scheduling precedence-constraint tasks on heterogeneous systems and the Cellular Processing Algorithms.

2. Problem Description

The High-Performance Computing systems (HPC) addressed in this paper consist of a set of heterogeneous machines M completely interconnected. Every machine $m_j \in M$ has different hardware characteristics, which yields different processing times for the same task. Without loss of generality, we make the following assumptions:

- Every machine has a connection link with any other machine.
- The communication links have no conflicts.
- All the communication links operate at the same speed.
- Every machine can send/receive information to/from another while executing a task.
- The communication cost between tasks in the same machine is depreciated.

2.1. Instance of the Problem

An instance of the problem is made up of two parts, a Directed Acyclic Graph (DAG) and the computation cost of the tasks in every machine. We represent the set of tasks of a parallel computing program and their precedencies as a DAG. Therefore, the parallel program is represented as the graph $G = (T, C)$ where T is the set of tasks (vertices) and C is the set of communication costs between tasks (edges) (see Figure 1). The complete instance of the problem is formed by G and the computational costs $P_{i,j}$ of each task t_i in every machine m_j (see Table 1).

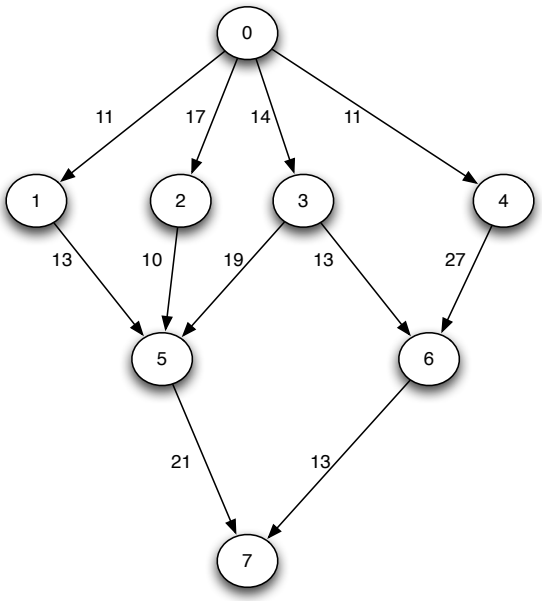


Figure 1. Graph $G = (T, C)$ of precedence tasks with communication costs $C_{i,j}$.

Table 1. Computational costs of the tasks $P_{i,j}$ on every machine.

Task	m_1	m_2	m_3
t_0	11	13	9
t_1	10	15	11
t_2	9	12	14
t_3	12	16	10
t_4	15	11	19
t_5	13	9	5
t_6	11	15	13
t_7	11	15	10

Any task t_i cannot be initiated until its precedent tasks $t_j \in T | (t_j, t_i) \in C$ finalize their executions and communications $(C_{j,i})$. However, when any pair of tasks are scheduled in the same machine, the actual communication cost $C'_{j,i}$ between them is depreciated.

2.2. Objective Function

In this work, we follow the approach of list scheduling algorithms. List scheduling algorithms are a family of heuristics in which tasks are ordered according to a particular priority criterion. The task execution order is equivalent to a DAG $G = (T, C)$ topological order from the task graph, without violating the precedence constraints. Table 2 shows an example of a feasible order for the task graph from Figure 1.

Table 2. Tasks' order of execution.

Task	t_0	t_4	t_3	t_1	t_2	t_5	t_6	t_7

Although the task order of execution is not indispensable for the scheduling, it simplifies the objective function computation [1], i.e., because it is not necessary to compute different combinations of the tasks, starting and finish times, to compute the minimum makespan [3,15,17]. However, this approach has the deficiency that the optimal value may not be possible in every task execution order. Algorithm 1 details the computation of the makespan objective function, using the computation times $P_{i,j}$ from Table 1 and a time counter ($Time_j$) for each machine, to keep track of the last executed task in each machine.

Algorithm 1 Makespan objective function

Input: $G = (T, C)$, computational costs $P_{t_i,j}$, and an order execution of the tasks $O = \{o_1, \ldots, o_{|T|}\}$.
Output: *makespan*

1: $Time_j \leftarrow 0, \forall m_j \in M$
2: **for** $x = 1$ to $|O|$ **do**
3: $t_{current} = o_x$
4: $j \leftarrow$ the index of the machine m_j assign to $t_{current}$
5: **if** $\forall u = \{1, ..., |T|\} \nexists (t_u, t_{current}) \in C$ **then**
6: $ts_{current} \leftarrow Time_j$
7: $tf_{current} \leftarrow ts_{current} + P_{t_{current},j}$
8: $Time_j \leftarrow tf_{current}$
9: **else**
10: $t_{u^*} \leftarrow \text{argmax}_{\{t_u | (t_u, t_{current}) \in C\}} \ (tf_u + C'_{t_u, t_{current}})$
11: $ts_{current} \leftarrow \max(tf_{u^*} + C'_{t_{u^*}, t_{current}}, Time_j)$
12: $tf_{current} \leftarrow ts_{current} + P_{t_{current},j}$
13: $Time_j \leftarrow tf_{current}$
14: **end if**
15: **end for**
16: **return** $makespan \leftarrow \max\{tf_i\} \forall t_i \in T$

Finally, the parallel program makespan (computation time) is the difference between the start and the ending of the first and last tasks. The makespan objective function uses the auxiliary variables ts_i (the starting time of task i), tf_i (the finish time of task i), and $C'_{i,j}$ which is the communication cost that is zero if the tasks are executed in the same machine. The objective is to compute the tasks, from the first to the last of the feasible execution order. Finally, the parallel program makespan (computation time) is the difference between the start and end of the first and last tasks. The complexity of the Algorithm 1 is $O(|T| \cdot |C|)$, although, in practice, it is remarkably lower than that, because not all the edges in G are connected to every node.

3. Algorithms Descriptions

This section introduces the generic metaheuristic frameworks (Sections 3.1 and 3.2), as well as a high-performance algorithm in the state-of-the-art (Section 3.3). Finally, our proposed algorithm is detailed in Section 3.4.

3.1. Iterated Local Search (ILS)

The ILS is a multi-start metaheuristic search based on local improvements (*LocalSearch*) and solution alterations (*Perturbation*) [31], see Algorithm 2. This algorithm starts initializing the current solution s' with a random solution, which is also assigned to the best solution s_{best} (see lines 1 and 2). The main loop of the algorithm iterates over the solution s' applying a perturbation followed by a Local

Search [32], if the ILS detects a new best-known solution, then s_{best} is updated (see line 7). The above process continues until the stopping criterion is reached, usually a maximum Central Processing Unit (CPU) time or a fixed number of objective function evaluations.

Algorithm 2 Iterated Local Search.

Input: Problem to solve
Output: s_{best}
1: $s' \leftarrow$ Random initial solution
2: $s_{best} \leftarrow s'$
3: **while** Stopping criterion not reached **do**
4: $s' \leftarrow Perturbation(s')$
5: $s' \leftarrow LocalSearch(s')$
6: **if** $f(s') < f(s_{best})$ **then**
7: $s_{best} \leftarrow s'$
8: **end if**
9: **end while**

3.2. Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP is a multi-start metaheuristic algorithm that builds a solution by selecting one promising fragment of the solution at a time [33], see Algorithm 3. The inner loop of the GRASP, in line 5 builds a *Solution* by adding random individual elements from a Restricted Candidate List (*RCL*) (see line 11). In order to build the *RCL*, the algorithm evaluates the increase of the partial objective and stores its maximum and minimum values (see lines 7 and 8) to set a limit for the original candidate list (*CL*) (see line 9), thus creating the *RCL* (see line 10); this process occurs at every step of the construction.

Algorithm 3 Greedy Randomized Adaptive Search Procedure.

Input: Problem to solve
Output: s_{best}
1: $s_{best} \leftarrow$ Random initial solution
2: **while** Stopping criterion not reached **do**
3: $Solution \leftarrow \emptyset$ ▷ An empty initial solution
4: $i = 0$
5: **while** *Solution* is not complete **do**
6: $CL \leftarrow SelectFeasibleElements()$
7: $f_{max} \leftarrow \max(PartialObjectiveEvaluation(t_i) \forall t_i \in CL)$
8: $f_{min} \leftarrow \min(PartialObjectiveEvaluation(t_i) \forall t_i \in CL)$
9: $limit \leftarrow f_{min} + \alpha(f_{max} - f_{min})$
10: $RCL \leftarrow BuildRCL(CL, limit)$
11: $Solution_i = t_r | t_r \in RCL$ ▷ Add to *Solution* a random element from the *RCL*
12: $i = i + 1$
13: **end while**
14: $Solution \leftarrow LocalSearch(Solution)$ ▷ The Local Search procedure is optional
15: **if** $f(\text{Solution}) < f(s_{best})$ **then**
16: $s_{best} \leftarrow Solution$
17: **end if**
18: **end while**

The *RCL* only includes candidate tasks whose incremental costs are bounded by $f_{min} + \alpha(f_{max} - f_{min})$ in line 9. Where f_{max} and f_{min} are the maximum and minimum incremental cost of the objective function, for all the candidate elements $t_i \in CL$, which is calculated with a modification of Algorithm 1 named *PartialObjectiveEvaluation* that evaluates up to the last element in the partial

solution. Additionally, $\alpha \in [0,1]$ defines the greedy level of the algorithm; where $\alpha = 0$ defines a completely greedy search, and $\alpha = 1$ defines a completely random search. Additionally, the candidate list (CL) must be created or updated every iteration of the inner loop (see line 6). Once the *Solution* is constructed, an optional *LocalSearch* procedure can be used to improve the current *Solution* (see line 14). Furthermore, s_{best} is updated every time a complete *Solution* outperforms the objective value of s_{best} (see line 16). Finally, the outer loop in line 2 restarts the *Solution* and iterates until the algorithm reaches the stopping criterion.

3.3. State-of-the-Art (Earliest Finish Time) EFT-ILS

In [11], the authors introduce an ILS, which will be called Earliest Finish Time (EFT)-ILS (in this paper), see Algorithm 4. EFT-ILS consists of two phases; the first explores random feasible execution orders of the tasks' graph from lines 3 to 10. After each new ordering o', the algorithm assigns the machines to the tasks that produce their Earliest Finish Time (EFT) (see line 5 and Algorithm 5). For heavy computational instances, we suggest using an external stopping criterion as CPU time, see line 10. The second phase initializes an ILS described in Section 3.1, using the best order o_{best} and solution s_{best}, found by the first phase, as initial solution. The next subsection details the Local Search and perturbation processes.

Algorithm 4 Earliest Finish Time-Iterated Local Search.

Input: $G = (T, C)$, computational costs $P_{i,j}$.
Output: s_{best}, o_{best}
1: $s' \leftarrow$ Random initial solution
2: $s_{best} \leftarrow s'$
3: **for** $i = 1$ to $MaxIterations$ **do**
4: $o' \leftarrow$ Random DAG G topological order
5: $s' \leftarrow EFT(o')$
6: **if** $f(s') < f(s_{best})$ **then**
7: $s_{best} \leftarrow s'$
8: $o_{best} \leftarrow o'$
9: **end if**
10: **end for** ▷ If the external stopping criterion is reach stop the for loop
11: $ILS(s_{best}, o_{best})$

Algorithm 5 Earliest Finish Time Function

Input: An order execution of the tasks $O = \{o_1, ..., o_{|T|}\}$
Output: An assignation of machines to tasks
1: **for** $i = 1$ to $|T|$ **do**
2: Assign to the task o_i the machine m_j which produce their minimum finish time.
3: **end for**

3.3.1. EFT-ILS Local Search

The Local Search (LS) in EFT-ILS is based on the first improvement pivoting rule, see Algorithm 6. This algorithm evaluates the tasks in the execution order O (see line 2). The algorithm generates neighbors s' assigning machines $m_j \in M$ to the current task $t_{current}$, if a neighbor improves solution s then s is updated (see line 7), as consequence the search is reinitialized in line 8. Finally, the algorithm verifies an auxiliary external stopping criterion before continuing with the neighbor generation process to avoid exceeding the maximum CPU time or objective function evaluations.

Algorithm 6 EFT-ILS Local Search procedure.

Input: Solution to improve s, and an order execution of the tasks $O = \{o_1, ..., o_{|T|}\}$
Output: s

```
1:  for i = 1 to |T| do
2:      t_current ← o_i                ▷ Assigns the task o_i in the execution order as the current task
3:      for j = 1 to |M| do
4:          s' ← s
5:          Assign t_current in s' to the machine m_j
6:          if f(s') < f(s) then
7:              s ← s'
8:              i = 1, j = 1
9:          end if
10:     end for                        ▷ If the external stopping criterion is reach stop the Local Search
11: end for
```

3.3.2. EFT-ILS Perturbation

EFT-ILS uses a perturbation process based on a probability. Every task t_i of the solution has a probability to be changed from its current machine. If the probability occurs the task t_i will be moved from its current machine and assigned to a new random one. For our experiment, we use a probability of 5% which is the best probability presented in [11].

3.4. Proposed GRASP-Cellular Processing Algorithm (GRASP-CPA)

In a similar manner as in EFT-ILS, our proposed algorithm GRASP-CPA consists of two phases (see Figure 2). First, a GRASP explores feasible tasks' orders for the next phase of the algorithm. In the second phase, the algorithm uses the best order o_{best} and solution s_{best} found by GRASP, in a homogeneous Cellular Processing Algorithm (CPA). The algorithm is composed by three ILS Processing Cells (PCells), the PCells have two functions that are independent from their ILS procedure. The first is to update the global best solution s_{best} if the PCell finds a better solution. The second is to update their current solutions through the communication processes. The communication is performed using the well-known single point-crossover from Genetic Algorithms (GAs), where two solutions from different PCells split and combine their information [12,34]. This phase continues until a fixed number of iterations or CPU seconds is reached.

Algorithm 7 describes our GRASP-CPA proposal, where the *GRASPConstruction* produces feasible orderings o' that are evaluated using *EFT* to produce the solution s' (see lines 5 and 6). Here, we can see that the algorithm uses a *GRASPConstruction* that receives an α value that can be either 0.9 or 1 with the same probability. This α value is used to restrict the candidate list (see lines 9 and 10 from Algorithm 3). GRASP algorithms usually use α values between 0.1 and 0.3. However, preliminary experimentation proved that the our proposed α values where the ones with the best performance for the instances used. Finally, line 12 executes the cellular processing section of the algorithm.

Algorithm 8 shows the general idea of the $CPA(s_{best}, o_{best})$ function. Here, the execution of the ILS Processing Cells (see lines 5 and 7) iterate five times each one, which limits the inner computational effort of the Processing Cells. After the Processing Cells' execution, the *Communication* processes the current solutions, recombining the s_{best} of PCell1 with the s_{best} of PCell2, the first offspring becomes the new current solution in PCell1. At the same time, the second offspring is used for a second recombination with the s_{best} of PCell3. The resulting offsprings from the second recombination become the new current solution of PCell2 and PCell3. This process continues until the stopping criterion is reached (see line 4).

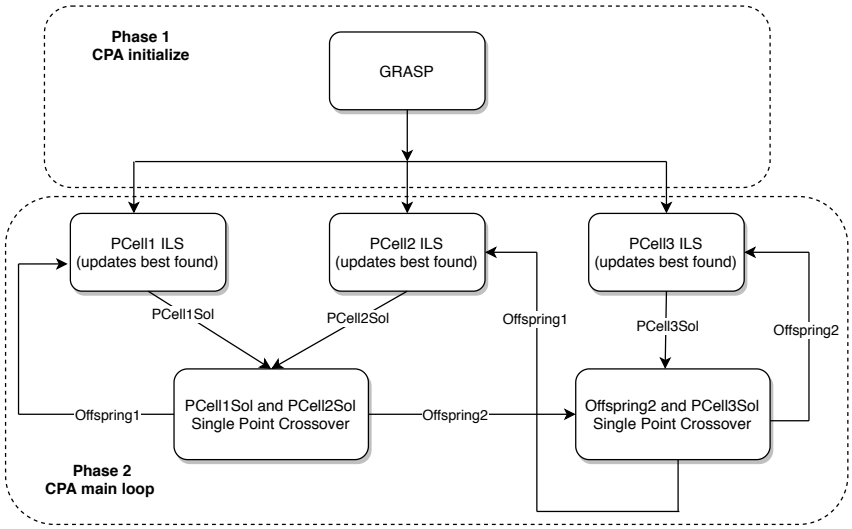


Figure 2. Proposed Greedy Randomized Adaptive Search Procedure (GRASP)-Cellular Processing Algorithm (CPA) flow chart diagram.

Algorithm 7 GRASP-CPA

Input: $G = (T, C)$, computational costs $P_{i,j}$.
Output: s_{best}, o_{best}

$s' \leftarrow$ Random initial solution
$s_{best} \leftarrow s'$
for $i = 1$ to $MaxIterations$ **do**
    $\alpha \leftarrow randomFrom(0.9, 1.0)$
    $o' \leftarrow GRASPConstruction(\alpha)$
    $s' \leftarrow EFT(o')$.
    if $f(s') < f(s_{best})$ **then**
        $s_{best} \leftarrow s'$
        $o_{best} \leftarrow o'$
    end if
end for ▷ If the external stopping criterion is reach stop the for loop
$CPA(s_{best}, o_{best})$.

Algorithm 8 Cellular Processing Algorithm section of the algorithm (GRASP-CPA)

Input: s_{best}, o_{best}
Output: s_{best}

$PCell1_{s_{current}} \leftarrow s_{best}$
$PCell2_{s_{current}} \leftarrow s_{best}$
$PCell3_{s_{current}} \leftarrow s_{best}$
while stopping criterion not reached **do**
    $PCell1ILS(PCell1_{s_{current}})$
    $PCell2ILS(PCell2_{s_{current}})$
    $PCell3ILS(PCell3_{s_{current}})$
    $Communication(PCell1_{s_{current}}, PCell2_{s_{current}}, PCell3_{s_{current}})$
end while

4. Experimental Setup

This section describes the experimental set of instances, the experimental configuration, and the statistical indicators of confidence in the results.

4.1. Parallel Application Instances

The applications used in the experimentation are:

- Double precision floating point FORTRAN benchmark (Fpppp) [35].
- The Laser Interferometer Gravitational-Wave Observatory (LIGO) application [36].
- Robot control application (Robot)
- Sparse matrix solver (Sparse).
- A benchmark of fourteen small synthetic instances from [11].

The applications Fpppp, Robot, and Sparse are included in the Standard Task Graph Set (STG) in [37]. We performed the same treatment to the original applications instances as in [8], considering different (Communication to Computation Ratios, CCRs) CCRs $= \{0.1, 0.5, 1, 5, 10\}$, (Heterogeneity Factors) HFs $= \{0.1, 0.25, 0.5, 0.75, 1\}$, and number of machines $|M| = \{8, 16, 32, 64\}$. The combination of the mentioned configurations for the four parallel applications gives a total of $4 \cdot 5 \cdot 5 \cdot 4 = 400$ instances. The nomenclature for the large instance set of Fpppp, Robot, and Sparse instances used in this work is Application-Machines-Tasks-HF-CCR. For the small benchmark in [11], the nomenclature is Name-Machines-Tasks. We made available the complete instance set in [38].

4.2. Experimental Settings

In the case of EFT-ILS, we use the best configuration in [11] (see Table 3). The GRASP-CPA uses a few extra parameters. One of them is the α value of the GRASP algorithm, for which we carried out extensive experimentation with $\alpha = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, finding that the best values for these instances were 1.0 and 0.9 without a clear dominance. Thus, we decide to use 1.0 and 0.9 as α values randomly with an equal probability before the *GRASPConstruction* (see line 5 of Algorithm 5). We use the same Local Search, perturbation process, and perturbation probability p_m in both algorithms for a fair comparison. Finally, to ensure the CPA communication process, we set the probability of recombination p_r as 100%. In both algorithms, the first phase has two stopping criteria, 50 iterations or a maximum of 5 min, while the second phase uses a stopping criterion of 100,000 objective function evaluations. Every instance from Section 4.1 is run 100 independent times. The complete experimental parameter settings are shown in Table 3. We made available the algorithm implementations at [39,40].

Table 3. Parameter settings for EFT-ILS and GRASP-CPA.

	EFT-ILS	**GRASP-CPA**
Phase 1 stopping criterion:	50 iterations/5 min	50 iterations/5 min
Phase 2 stopping criterion:	100,000 MaxEvaluations	100,000 MaxEvaluations
α:	-	1.0 and 0.9
Communication:	-	single-point cx, $p_r = 1.0$
ILS max iterations:	unlimited/MaxEvaluations	5/MaxEvaluations
Perturbation:	Section 3.3.2, $P_m = 0.05$	Section 3.3.2, $P_m = 0.05$
Local Search	Algorithm 6	Algorithm 6

4.3. Statistical Indicators

To assess statistical confidence in our experimentation, we compute the median value from the independent runs as well as the interquartile range (IQR). The tables presenting our results follow the format $MEDIAN_{IQR}$. The tables also emphasize the best and second-best reported values for every

problem with a gray and light background, respectively. For the sake of completeness, we apply the non-parametric Wilcoxon signed ranks test on the results to assess statistical differences in a pairwise comparison for every problem, at 95% confidence level [41]. A symbol ▲ indicates that EFT-ILS was statistically worse than GRASP-CPA according to the Wilcoxon signed ranks test; we use ▽ otherwise. Finally, we marked as '–' the cases where there were no statistical differences.

5. Results

First, we start analyzing the results of the large benchmark set of 400 scheduling problems. Focusing on the Fpppp instance set (see Table 4), GRASP-CPA outperformed with statistical significance EFT-ILS in 53 instances, not statistically outperforming in any instance with 8 and 16 machines. However, for the instances with 32 and 64 machines, EFT-ILS outperformed GRASP-CPA in 3 and 11 instances, respectively.

Regarding the LIGO benchmark results from Table 5, GRASP-CPA outperformed EFT-ILS in 45 instances. EFT-ILS only outperformed GRASP-CPA in 13 instances, distributed as follows: two, three, five, and three for the instances with 8, 16, 32, and 64 machines, respectively.

For the Robot benchmark (see Table 6), GRASP-CPA outperformed EFT-ILS with statistical significance in 48 cases, while EFT-ILS only outperformed GRASP-CPA in 12 instances, where most of them occurred for the instances with 16 machines.

The results for the Sparse benchmark from Table 7 show that GRASP-CPA outperformed EFT-ILS with statistical significance in 33 instances, while EFT-ILS outperformed GRASP-CPA in 10 instances.

Finally, for these instances sets, GRASP-CPA achieved 265% more best median values found than EFT-ILS, with statistical significance. Therefore, we consider that GRASP-CPA is superior to EFT-ILS in a relevant proportion of the studied cases.

Furthermore, we analyze the results from the small benchmark of 14 synthetic instances in Table 8. For the 14 synthetic problems, GRASP-CPA achieves the best median value in all the cases, with statistical significance in ten cases. In addition, Table 8 shows the average computing time of the enumerative optimal algorithm ($\text{Time}_{\text{Enum}}$), EFT-ILS ($\text{Time}_{\text{EFT-ILS}}$), and GRASP-CPA ($\text{Time}_{\text{GRASP-CPA}}$) in CPU seconds on a Macbook pro 13-inch late 2011. Table 9 presents the best solution found by EFT-ILS and GRASP-CPA, where EFT-ILS computes twelve optimal values, while GRASP-CPA computes 13. Additionally, GRASP-CPA achieves an IQR of 0.0 in six cases where the algorithm reached the optimal value in the median.

Table 4. Fpppp instances median and IQR of EFT-ILS and GRASP-CPA over 100 independent runs. Light gray emphasizes the best results.

Problem	EFT-ILS	GRASP-CPA	Problem	EFT-ILS	GRASP-CPA
Fpppp-8-334-0.1-0.1	$9.40E1_{1.1E0}-$	$9.39E1_{1.2E0}$	Fpppp-32-334-0.1-0.1	$2.28E2_{8.4E0}-$	$2.28E2_{6.8E0}$
Fpppp-8-334-0.1-0.5	$6.05E2_{9.8E0}\blacktriangle$	$6.00E2_{8.7E0}$	Fpppp-32-334-0.1-0.5	$8.88E2_{2.1E1}\triangledown$	$8.93E2_{3.2E1}$
Fpppp-8-334-0.1-1	$1.49E3_{2.5E1}\blacktriangle$	$1.48E3_{2.7E1}$	Fpppp-32-334-0.1-1	$1.12E3_{3.0E1}\blacktriangle$	$1.12E3_{2.5E1}$
Fpppp-8-334-0.1-5	$2.65E3_{9.2E1}\blacktriangle$	$2.60E3_{8.8E1}$	Fpppp-32-334-0.1-5	$1.55E3_{7.0E1}\blacktriangle$	$1.52E3_{7.6E1}$
Fpppp-8-334-0.1-10	$2.12E3_{7.7E1}\blacktriangle$	$2.05E3_{9.7E1}$	Fpppp-32-334-0.1-10	$5.66E3_{2.7E2}\blacktriangle$	$5.51E3_{3.2E2}$
Fpppp-8-334-0.25-0.1	$8.67E2_{1.1E1}\blacktriangle$	$8.61E2_{1.1E1}$	Fpppp-32-334-0.25-0.1	$8.44E2_{2.8E1}-$	$8.40E2_{3.3E1}$
Fpppp-8-334-0.25-0.5	$2.65E3_{3.7E1}\blacktriangle$	$2.64E3_{3.7E1}$	Fpppp-32-334-0.25-0.5	$5.71E2_{1.4E1}\triangledown$	$5.75E2_{1.6E1}$
Fpppp-8-334-0.25-1	$1.94E3_{3.0E1}\blacktriangle$	$1.93E3_{3.3E1}$	Fpppp-32-334-0.25-1	$2.66E2_{8.5E0}-$	$2.67E2_{6.8E0}$
Fpppp-8-334-0.25-5	$5.86E2_{2.1E1}\blacktriangle$	$5.73E2_{2.4E1}$	Fpppp-32-334-0.25-5	$2.76E3_{1.5E2}\blacktriangle$	$2.70E3_{1.3E2}$
Fpppp-8-334-0.25-10	$2.82E3_{1.2E2}\blacktriangle$	$2.70E3_{1.3E2}$	Fpppp-32-334-0.25-10	$4.42E3_{1.6E2}-$	$4.41E3_{1.7E2}$
Fpppp-8-334-0.5-0.1	$3.87E2_{4.7E0}\blacktriangle$	$3.83E2_{5.6E0}$	Fpppp-32-334-0.5-0.1	$1.04E3_{3.6E1}-$	$1.03E3_{3.4E1}$
Fpppp-8-334-0.5-0.5	$2.25E3_{3.9E1}\blacktriangle$	$2.23E3_{3.8E1}$	Fpppp-32-334-0.5-0.5	$1.23E3_{4.1E1}\triangledown$	$1.24E3_{4.2E1}$
Fpppp-8-334-0.5-1	$2.07E3_{4.4E1}\blacktriangle$	$2.06E3_{4.2E1}$	Fpppp-32-334-0.5-1	$6.64E2_{1.8E1}-$	$6.66E2_{1.6E1}$
Fpppp-8-334-0.5-5	$4.23E3_{1.4E2}\blacktriangle$	$4.08E3_{1.3E2}$	Fpppp-32-334-0.5-5	$2.41E3_{1.5E2}-$	$2.41E3_{1.4E2}$
Fpppp-8-334-0.5-10	$2.73E3_{1.3E2}\blacktriangle$	$2.60E3_{1.5E2}$	Fpppp-32-334-0.5-10	$3.08E3_{1.2E2}\blacktriangle$	$3.03E3_{1.1E2}$
Fpppp-8-334-0.75-0.1	$3.00E2_{5.8E0}\blacktriangle$	$2.96E2_{4.8E0}$	Fpppp-32-334-0.75-0.1	$8.78E2_{3.1E1}\blacktriangle$	$8.71E2_{3.1E1}$
Fpppp-8-334-0.75-0.5	$3.93E2_{8.4E0}\blacktriangle$	$3.89E2_{6.0E0}$	Fpppp-32-334-0.75-0.5	$1.28E3_{2.7E1}-$	$1.28E3_{2.8E1}$
Fpppp-8-334-0.75-1	$7.64E2_{1.7E1}\blacktriangle$	$7.53E2_{1.4E1}$	Fpppp-32-334-0.75-1	$7.89E2_{2.3E1}-$	$7.97E2_{1.9E1}$
Fpppp-8-334-0.75-5	$2.87E3_{1.3E2}\blacktriangle$	$2.77E3_{1.4E2}$	Fpppp-32-334-0.75-5	$8.48E2_{3.1E1}\blacktriangle$	$8.38E2_{2.4E1}$
Fpppp-8-334-0.75-10	$4.92E3_{1.9E2}\blacktriangle$	$4.74E3_{2.4E2}$	Fpppp-32-334-0.75-10	$2.54E3_{7.0E1}\blacktriangle$	$2.51E3_{7.3E1}$
Fpppp-8-334-1-0.1	$7.97E2_{1.1E1}\blacktriangle$	$7.86E2_{1.0E1}$	Fpppp-32-334-1-0.1	$6.90E2_{2.2E1}-$	$6.90E2_{2.4E1}$
Fpppp-8-334-1-0.5	$1.31E3_{3.3E1}\blacktriangle$	$1.30E3_{2.3E1}$	Fpppp-32-334-1-0.5	$6.28E2_{2.3E1}-$	$6.29E2_{1.8E1}$
Fpppp-8-334-1-1	$3.21E2_{7.3E0}\blacktriangle$	$3.16E2_{8.4E0}$	Fpppp-32-334-1-1	$6.81E1_{1.7E0}-$	$6.82E1_{1.8E0}$
Fpppp-8-334-1-5	$4.63E3_{2.0E2}\blacktriangle$	$4.53E3_{2.2E2}$	Fpppp-32-334-1-5	$2.13E3_{1.1E2}-$	$2.13E3_{8.0E1}$
Fpppp-8-334-1-10	$3.35E3_{1.4E2}\blacktriangle$	$3.21E3_{1.0E2}$	Fpppp-32-334-1-10	$4.46E3_{2.5E2}\blacktriangle$	$4.34E3_{1.9E2}$
Fpppp-16-334-0.1-0.1	$3.70E2_{7.7E0}-$	$3.72E2_{9.6E0}$	Fpppp-64-334-0.1-0.1	$2.60E2_{1.7E0}\blacktriangle$	$2.60E2_{1.4E0}$
Fpppp-16-334-0.1-0.5	$1.93E3_{6.5E1}-$	$1.92E3_{7.8E1}$	Fpppp-64-334-0.1-0.5	$5.54E1_{8.4E-1}\triangledown$	$5.58E1_{7.8E-1}$
Fpppp-16-334-0.1-1	$8.72E2_{2.7E1}\blacktriangle$	$8.66E2_{3.4E1}$	Fpppp-64-334-0.1-1	$7.31E2_{1.1E1}\triangledown$	$7.36E2_{1.0E1}$
Fpppp-16-334-0.1-5	$3.26E3_{1.2E2}-$	$3.25E3_{1.2E2}$	Fpppp-64-334-0.1-5	$2.00E3_{1.0E2}\triangledown$	$2.03E3_{9.2E1}$
Fpppp-16-334-0.1-10	$5.57E3_{2.4E2}\blacktriangle$	$5.37E3_{3.0E2}$	Fpppp-64-334-0.1-10	$2.26E3_{1.1E2}\triangledown$	$2.28E3_{1.3E2}$
Fpppp-16-334-0.25-0.1	$1.80E3_{4.2E1}\blacktriangle$	$1.78E3_{4.5E1}$	Fpppp-64-334-0.25-0.1	$8.93E2_{5.2E0}\triangledown$	$8.97E2_{6.1E0}$
Fpppp-16-334-0.25-0.5	$1.78E3_{5.7E1}-$	$1.78E3_{6.0E1}$	Fpppp-64-334-0.25-0.5	$1.33E3_{1.4E1}\triangledown$	$1.34E3_{1.6E1}$
Fpppp-16-334-0.25-1	$5.15E2_{2.5E1}-$	$5.12E2_{1.6E1}$	Fpppp-64-334-0.25-1	$9.48E2_{2.0E1}-$	$9.52E2_{2.6E1}$
Fpppp-16-334-0.25-5	$1.87E3_{7.0E1}\blacktriangle$	$1.85E3_{7.9E1}$	Fpppp-64-334-0.25-5	$2.06E3_{9.9E1}-$	$2.06E3_{9.6E1}$
Fpppp-16-334-0.25-10	$8.07E2_{3.3E1}\blacktriangle$	$7.79E2_{3.1E1}$	Fpppp-64-334-0.25-10	$1.74E3_{7.6E1}-$	$1.75E3_{8.1E1}$
Fpppp-16-334-0.5-0.1	$1.46E3_{3.0E1}\blacktriangle$	$1.44E3_{2.6E1}$	Fpppp-64-334-0.5-0.1	$2.04E2_{3.3E-1}\blacktriangle$	$2.04E2_{1.5E0}$
Fpppp-16-334-0.5-0.5	$1.35E3_{3.6E1}-$	$1.35E3_{4.0E1}$	Fpppp-64-334-0.5-0.5	$1.13E3_{1.1E1}\blacktriangle$	$1.13E3_{1.2E1}$
Fpppp-16-334-0.5-1	$3.80E2_{1.4E1}\blacktriangle$	$3.74E2_{1.1E1}$	Fpppp-64-334-0.5-1	$5.96E2_{1.1E1}\blacktriangle$	$6.01E2_{7.4E0}$
Fpppp-16-334-0.5-5	$2.82E3_{1.7E2}-$	$2.85E3_{1.3E2}$	Fpppp-64-334-0.5-5	$1.23E3_{6.1E1}\blacktriangle$	$1.26E3_{6.0E1}$
Fpppp-16-334-0.5-10	$9.22E2_{2.9E1}\blacktriangle$	$8.94E2_{3.4E1}$	Fpppp-64-334-0.5-10	$1.61E3_{6.8E1}-$	$1.62E3_{5.4E1}$
Fpppp-16-334-0.75-0.1	$3.27E2_{7.2E0}\blacktriangle$	$3.24E2_{9.2E0}$	Fpppp-64-334-0.75-0.1	$1.18E2_{8.0E-2}\blacktriangle$	$1.18E2_{6.7E-1}$
Fpppp-16-334-0.75-0.5	$5.87E2_{2.2E1}-$	$5.85E2_{1.7E1}$	Fpppp-64-334-0.75-0.5	$4.72E2_{9.2E0}\triangledown$	$4.76E2_{1.1E1}$
Fpppp-16-334-0.75-1	$6.12E2_{2.2E1}-$	$6.12E2_{2.0E1}$	Fpppp-64-334-0.75-1	$4.98E2_{1.2E1}-$	$5.00E2_{1.1E1}$
Fpppp-16-334-0.75-5	$3.73E3_{2.6E2}\blacktriangle$	$3.68E3_{1.9E2}$	Fpppp-64-334-0.75-5	$8.96E2_{4.9E1}-$	$9.06E2_{5.3E1}$
Fpppp-16-334-0.75-10	$1.99E3_{1.0E2}\blacktriangle$	$1.92E3_{8.2E1}$	Fpppp-64-334-0.75-10	$3.44E3_{1.4E2}-$	$3.46E3_{1.4E2}$
Fpppp-16-334-1-0.1	$7.48E2_{1.8E1}\blacktriangle$	$7.40E2_{1.7E1}$	Fpppp-64-334-1-0.1	$8.22E1_{5.2E-1}\triangledown$	$8.26E1_{6.8E-1}$
Fpppp-16-334-1-0.5	$9.56E2_{2.7E1}-$	$9.47E2_{2.8E1}$	Fpppp-64-334-1-0.5	$3.55E2_{3.8E0}\triangledown$	$3.56E2_{2.5E0}$
Fpppp-16-334-1-1	$4.58E2_{1.3E1}\blacktriangle$	$4.54E2_{1.5E1}$	Fpppp-64-334-1-1	$1.28E3_{1.2E1}\triangledown$	$1.29E3_{1.4E1}$
Fpppp-16-334-1-5	$1.45E3_{6.4E1}-$	$1.44E3_{6.8E1}$	Fpppp-64-334-1-5	$8.50E2_{5.1E1}-$	$8.58E2_{5.8E1}$
Fpppp-16-334-1-10	$9.84E2_{3.5E1}\blacktriangle$	$9.54E2_{4.2E1}$	Fpppp-64-334-1-10	$4.34E3_{1.6E2}\triangledown$	$4.39E3_{1.8E2}$

▲ GRASP-CPA outperforms EFT-ILS with statistical difference, otherwise ▽, − no statistical difference.

Table 5. LIGO instances median and IQR of EFT-ILS and GRASP-CPA over 100 independent runs. Light gray emphasizes the best results.

Problem	EFT-ILS	GRASP-CPA	Problem	EFT-ILS	GRASP-CPA
LIGO-8-76-0.1-0.1	$6.10E2_{1.9E1}$▲	$6.08E2_{1.4E1}$	LIGO-32-76-0.1-0.1	$5.40E2_{1.4E0}$▽	$5.41E2_{1.3E0}$
LIGO-8-76-0.1-0.5	$9.24E2_{1.8E1}$−	$9.20E2_{2.0E1}$	LIGO-32-76-0.1-0.5	$6.00E2_{1.2E1}$▽	$6.02E2_{1.3E1}$
LIGO-8-76-0.1-1	$1.04E3_{2.5E1}$▲	$1.03E3_{3.8E1}$	LIGO-32-76-0.1-1	$3.03E1_{8.5E-1}$−	$3.03E1_{8.3E-1}$
LIGO-8-76-0.1-5	$1.36E3_{5.2E1}$▲	$1.30E3_{7.1E1}$	LIGO-32-76-0.1-5	$1.30E3_{6.3E1}$▲	$1.26E3_{8.7E1}$
LIGO-8-76-0.1-10	$1.48E3_{7.5E1}$▲	$1.46E3_{1.1E2}$	LIGO-32-76-0.1-10	$2.01E3_{1.2E2}$▲	$1.96E3_{1.1E2}$
LIGO-8-76-0.25-0.1	$4.91E2_{1.4E1}$▲	$4.82E2_{1.1E1}$	LIGO-32-76-0.25-0.1	$1.03E2_{8.1E-1}$−	$1.03E2_{7.2E-1}$
LIGO-8-76-0.25-0.5	$7.54E2_{2.4E1}$▲	$7.49E2_{1.8E1}$	LIGO-32-76-0.25-0.5	$8.64E1_{1.5E0}$▲	$8.59E1_{2.4E0}$
LIGO-8-76-0.25-1	$4.76E2_{1.7E1}$▲	$4.70E2_{1.2E1}$	LIGO-32-76-0.25-1	$8.43E2_{2.2E1}$−	$8.42E2_{2.7E1}$
LIGO-8-76-0.25-5	$2.15E2_{8.1E0}$▲	$2.11E2_{9.3E0}$	LIGO-32-76-0.25-5	$5.86E2_{2.0E1}$−	$5.82E2_{2.3E1}$
LIGO-8-76-0.25-10	$1.74E3_{1.2E2}$▲	$1.71E3_{8.4E1}$	LIGO-32-76-0.25-10	$8.25E2_{5.9E1}$▲	$8.06E2_{6.4E1}$
LIGO-8-76-0.5-0.1	$4.30E2_{1.1E1}$▲	$4.24E2_{1.1E1}$	LIGO-32-76-0.5-0.1	$6.76E2_{0.0E0}$−	$6.76E2_{0.0E0}$
LIGO-8-76-0.5-0.5	$8.17E2_{2.4E1}$▲	$8.05E2_{2.6E1}$	LIGO-32-76-0.5-0.5	$3.72E2_{8.4E0}$▽	$3.77E2_{1.0E1}$
LIGO-8-76-0.5-1	$8.06E2_{2.6E1}$▽	$8.15E2_{3.0E1}$	LIGO-32-76-0.5-1	$4.22E2_{1.3E1}$−	$4.22E2_{1.3E1}$
LIGO-8-76-0.5-5	$1.77E3_{1.1E2}$−	$1.76E3_{8.6E1}$	LIGO-32-76-0.5-5	$6.40E2_{3.2E1}$▲	$6.33E2_{3.3E1}$
LIGO-8-76-0.5-10	$3.41E2_{2.6E1}$▽	$3.46E2_{2.3E1}$	LIGO-32-76-0.5-10	$1.70E3_{1.0E2}$▲	$1.63E3_{6.1E1}$
LIGO-8-76-0.75-0.1	$1.75E2_{3.7E0}$▲	$1.71E2_{4.5E0}$	LIGO-32-76-0.75-0.1	$5.70E2_{0.0E0}$−	$5.70E2_{0.0E0}$
LIGO-8-76-0.75-0.5	$5.19E2_{1.6E1}$▲	$5.09E2_{1.4E1}$	LIGO-32-76-0.75-0.5	$3.19E2_{1.4E0}$−	$3.19E2_{1.2E0}$
LIGO-8-76-0.75-1	$7.62E2_{2.6E1}$▲	$7.52E2_{2.0E1}$	LIGO-32-76-0.75-1	$6.60E1_{1.5E0}$▽	$6.61E1_{1.0E0}$
LIGO-8-76-0.75-5	$4.78E2_{2.6E1}$▲	$4.71E2_{2.5E1}$	LIGO-32-76-0.75-5	$1.42E3_{5.5E1}$▲	$1.39E3_{9.4E1}$
LIGO-8-76-0.75-10	$9.21E2_{4.2E1}$▲	$8.73E2_{6.3E1}$	LIGO-32-76-0.75-10	$1.46E3_{7.0E1}$▲	$1.44E3_{8.4E1}$
LIGO-8-76-1-0.1	$1.41E2_{3.8E0}$▲	$1.36E2_{3.0E0}$	LIGO-32-76-1-0.1	$4.23E2_{0.0E0}$−	$4.23E2_{0.0E0}$
LIGO-8-76-1-0.5	$4.13E2_{1.3E1}$−	$4.16E2_{1.1E1}$	LIGO-32-76-1-0.5	$1.09E2_{2.9E0}$▽	$1.10E2_{2.3E0}$
LIGO-8-76-1-1	$8.08E2_{2.0E1}$−	$8.09E2_{1.7E1}$	LIGO-32-76-1-1	$1.59E2_{3.5E0}$−	$1.59E2_{4.5E0}$
LIGO-8-76-1-5	$4.22E2_{2.3E1}$▲	$4.11E2_{2.1E1}$	LIGO-32-76-1-5	$5.48E2_{4.0E1}$▲	$5.33E2_{3.2E1}$
LIGO-8-76-1-10	$2.75E2_{1.5E1}$▲	$2.68E2_{1.2E1}$	LIGO-32-76-1-10	$2.25E3_{2.6E2}$▲	$2.14E3_{1.2E2}$
LIGO-16-76-0.1-0.1	$5.52E1_{5.0E-1}$▽	$5.54E1_{9.8E-1}$	LIGO-64-76-0.1-0.1	$6.02E2_{4.5E-1}$▲	$6.02E2_{1.1E0}$
LIGO-16-76-0.1-0.5	$5.87E2_{7.0E0}$▽	$5.89E2_{6.8E0}$	LIGO-64-76-0.1-0.5	$4.22E2_{1.5E1}$−	$4.17E2_{1.4E1}$
LIGO-16-76-0.1-1	$7.78E2_{3.6E1}$▽	$7.85E2_{3.3E1}$	LIGO-64-76-0.1-1	$1.72E2_{3.7E0}$▽	$1.73E2_{3.8E0}$
LIGO-16-76-0.1-5	$1.28E3_{8.7E1}$−	$1.28E3_{8.7E1}$	LIGO-64-76-0.1-5	$1.10E3_{3.9E1}$−	$1.11E3_{4.9E1}$
LIGO-16-76-0.1-10	$2.23E3_{1.5E2}$−	$2.23E3_{1.7E2}$	LIGO-64-76-0.1-10	$1.32E3_{7.7E1}$▲	$1.27E3_{6.8E1}$
LIGO-16-76-0.25-0.1	$2.94E2_{0.0E0}$▲	$2.94E2_{4.5E-1}$	LIGO-64-76-0.25-0.1	$5.92E1_{6.8E-1}$▽	$5.93E1_{8.5E-1}$
LIGO-16-76-0.25-0.5	$2.98E2_{1.1E0}$▲	$2.97E2_{1.9E0}$	LIGO-64-76-0.25-0.5	$1.69E2_{1.9E0}$−	$1.69E2_{2.4E0}$
LIGO-16-76-0.25-1	$1.01E2_{1.5E0}$−	$1.02E2_{1.7E0}$	LIGO-64-76-0.25-1	$4.27E2_{1.7E1}$▽	$4.30E2_{1.2E1}$
LIGO-16-76-0.25-5	$1.02E3_{5.2E1}$−	$1.03E3_{4.5E1}$	LIGO-64-76-0.25-5	$1.68E3_{7.5E1}$−	$1.69E3_{8.0E1}$
LIGO-16-76-0.25-10	$5.77E2_{3.7E1}$▲	$5.57E2_{2.1E1}$	LIGO-64-76-0.25-10	$1.81E3_{9.3E1}$▲	$1.78E3_{8.9E1}$
LIGO-16-76-0.5-0.1	$1.35E2_{1.8E-1}$−	$1.35E2_{2.4E-1}$	LIGO-64-76-0.5-0.1	$6.26E2_{0.0E0}$−	$6.26E2_{0.0E0}$
LIGO-16-76-0.5-0.5	$3.62E2_{8.8E0}$−	$3.62E2_{7.4E0}$	LIGO-64-76-0.5-0.5	$6.21E2_{5.6E0}$−	$6.22E2_{2.7E0}$
LIGO-16-76-0.5-1	$9.26E2_{4.1E1}$−	$9.19E2_{3.4E1}$	LIGO-64-76-0.5-1	$3.26E2_{8.1E0}$▲	$3.22E2_{1.2E1}$
LIGO-16-76-0.5-5	$2.80E2_{9.2E0}$−	$2.81E2_{8.6E0}$	LIGO-64-76-0.5-5	$1.25E3_{7.8E1}$−	$1.27E3_{8.9E1}$
LIGO-16-76-0.5-10	$2.45E2_{1.7E1}$−	$2.42E2_{1.2E1}$	LIGO-64-76-0.5-10	$9.80E2_{5.5E1}$▲	$9.47E2_{5.3E1}$
LIGO-16-76-0.75-0.1	$4.93E2_{0.0E0}$−	$4.93E2_{0.0E0}$	LIGO-64-76-0.75-0.1	$2.72E2_{0.0E0}$−	$2.72E2_{0.0E0}$
LIGO-16-76-0.75-0.5	$1.65E2_{5.6E0}$−	$1.65E2_{4.2E0}$	LIGO-64-76-0.75-0.5	$2.79E2_{0.0E0}$−	$2.79E2_{0.0E0}$
LIGO-16-76-0.75-1	$3.78E2_{1.1E1}$▲	$3.76E2_{1.4E1}$	LIGO-64-76-0.75-1	$3.33E2_{1.4E1}$▲	$3.27E2_{9.3E0}$
LIGO-16-76-0.75-5	$1.09E3_{5.0E1}$−	$1.10E3_{6.0E1}$	LIGO-64-76-0.75-5	$8.42E2_{6.0E1}$▲	$8.17E2_{5.0E1}$
LIGO-16-76-0.75-10	$6.77E2_{4.5E1}$−	$6.74E2_{3.6E1}$	LIGO-64-76-0.75-10	$2.42E3_{1.2E2}$▲	$2.33E3_{9.7E1}$
LIGO-16-76-1-0.1	$1.11E2_{5.2E-1}$▲	$1.11E2_{1.3E0}$	LIGO-64-76-1-0.1	$3.01E2_{0.0E0}$−	$3.01E2_{0.0E0}$
LIGO-16-76-1-0.5	$1.34E2_{1.8E0}$−	$1.34E2_{1.7E0}$	LIGO-64-76-1-0.5	$2.55E2_{0.0E0}$−	$2.55E2_{0.0E0}$
LIGO-16-76-1-1	$2.85E2_{6.7E0}$−	$2.85E2_{4.7E0}$	LIGO-64-76-1-1	$1.63E2_{2.1E0}$−	$1.63E2_{2.1E0}$
LIGO-16-76-1-5	$1.39E2_{6.0E0}$▲	$1.38E2_{6.0E0}$	LIGO-64-76-1-5	$1.50E3_{8.2E1}$−	$1.51E3_{7.5E1}$
LIGO-16-76-1-10	$2.24E2_{2.4E1}$▲	$2.29E2_{2.3E1}$	LIGO-64-76-1-10	$1.20E2_{7.8E0}$▲	$1.18E2_{8.3E0}$

Table 6. Robot instances median and IQR of EFT-ILS and GRASP-CPA over 100 independent runs. Light gray emphasizes the best results.

Problem	EFT-ILS	GRASP-CPA	Problem	EFT-ILS	GRASP-CPA
Robot-8-88-0.1-0.1	$1.89E3_{2.7E1}$▲	$1.88E3_{1.6E1}$	Robot-32-88-0.1-0.1	$1.07E3_{2.3E0}$−	$1.07E3_{2.3E0}$
Robot-8-88-0.1-0.5	$7.22E1_{1.3E0}$−	$7.20E1_{1.2E0}$	Robot-32-88-0.1-0.5	$1.07E3_{3.8E0}$▲	$1.06E3_{1.1E1}$
Robot-8-88-0.1-1	$2.02E3_{4.4E1}$▲	$1.98E3_{3.7E1}$	Robot-32-88-0.1-1	$1.32E3_{1.3E1}$▽	$1.33E3_{1.1E1}$
Robot-8-88-0.1-5	$3.03E3_{1.1E2}$−	$3.04E3_{1.0E2}$	Robot-32-88-0.1-5	$2.44E3_{9.8E1}$▲	$2.35E3_{1.0E2}$
Robot-8-88-0.1-10	$2.39E3_{1.0E2}$▲	$2.24E3_{9.0E1}$	Robot-32-88-0.1-10	$2.97E2_{1.1E1}$▲	$2.74E2_{1.1E1}$
Robot-8-88-0.25-0.1	$6.45E2_{1.1E1}$▲	$6.40E2_{9.3E0}$	Robot-32-88-0.25-0.1	$1.26E3_{3.0E0}$−	$1.26E3_{2.4E0}$
Robot-8-88-0.25-0.5	$6.52E2_{9.9E0}$▲	$6.50E2_{8.2E0}$	Robot-32-88-0.25-0.5	$8.73E1_{8.8E-1}$▲	$8.71E1_{8.9E-1}$
Robot-8-88-0.25-1	$4.71E2_{6.4E0}$▽	$4.73E2_{8.1E0}$	Robot-32-88-0.25-1	$1.91E3_{2.7E1}$−	$1.91E3_{2.6E1}$
Robot-8-88-0.25-5	$1.55E3_{4.5E1}$−	$1.56E3_{4.5E1}$	Robot-32-88-0.25-5	$3.05E3_{1.1E2}$−	$3.04E3_{8.9E1}$
Robot-8-88-0.25-10	$2.73E3_{9.6E1}$▲	$2.66E3_{8.2E1}$	Robot-32-88-0.25-10	$1.85E3_{8.2E1}$▲	$1.68E3_{7.6E1}$
Robot-8-88-0.5-0.1	$1.03E3_{1.6E1}$▲	$1.02E3_{1.3E1}$	Robot-32-88-0.5-0.1	$3.75E2_{2.7E0}$▲	$3.74E2_{7.7E-1}$
Robot-8-88-0.5-0.5	$7.93E2_{1.6E1}$▲	$7.85E2_{1.5E1}$	Robot-32-88-0.5-0.5	$3.23E2_{3.8E0}$▲	$3.21E2_{3.8E0}$
Robot-8-88-0.5-1	$7.45E2_{1.1E1}$▲	$7.41E2_{1.2E1}$	Robot-32-88-0.5-1	$8.58E2_{1.2E1}$▽	$8.65E2_{1.4E1}$
Robot-8-88-0.5-5	$1.35E3_{4.0E1}$▲	$1.34E3_{3.0E1}$	Robot-32-88-0.5-5	$7.28E2_{1.7E1}$−	$7.24E2_{1.8E1}$
Robot-8-88-0.5-10	$3.60E2_{1.8E1}$▲	$3.49E2_{1.4E1}$	Robot-32-88-0.5-10	$1.98E3_{7.2E1}$−	$1.99E3_{6.3E1}$
Robot-8-88-0.75-0.1	$5.64E2_{1.2E1}$▲	$5.60E2_{1.1E1}$	Robot-32-88-0.75-0.1	$9.42E2_{0.0E0}$−	$9.42E2_{1.5E0}$
Robot-8-88-0.75-0.5	$1.59E3_{2.1E1}$▲	$1.58E3_{1.7E1}$	Robot-32-88-0.75-0.5	$1.28E3_{1.5E1}$−	$1.28E3_{1.2E1}$
Robot-8-88-0.75-1	$1.40E3_{2.9E1}$−	$1.40E3_{2.2E1}$	Robot-32-88-0.75-1	$1.39E3_{1.3E1}$−	$1.39E3_{1.8E1}$
Robot-8-88-0.75-5	$2.52E3_{7.1E1}$▲	$2.41E3_{8.0E1}$	Robot-32-88-0.75-5	$1.50E3_{5.0E1}$▽	$1.51E3_{4.4E1}$
Robot-8-88-0.75-10	$1.09E3_{4.2E1}$▲	$1.04E3_{3.8E1}$	Robot-32-88-0.75-10	$4.75E2_{1.5E1}$▲	$4.56E2_{1.8E1}$
Robot-8-88-1-0.1	$3.04E2_{7.2E0}$▲	$2.96E2_{5.7E0}$	Robot-32-88-1-0.1	$1.02E3_{3.2E0}$−	$1.02E3_{3.2E0}$
Robot-8-88-1-0.5	$3.73E2_{9.3E0}$▲	$3.66E2_{8.0E0}$	Robot-32-88-1-0.5	$4.67E2_{0.0E0}$−	$4.67E2_{4.1E-1}$
Robot-8-88-1-1	$1.38E3_{4.0E1}$▲	$1.36E3_{4.3E1}$	Robot-32-88-1-1	$7.96E2_{2.3E1}$▲	$7.78E2_{3.2E1}$
Robot-8-88-1-5	$9.94E2_{2.8E1}$−	$9.95E2_{2.8E1}$	Robot-32-88-1-5	$1.16E3_{3.1E1}$▲	$1.14E3_{2.9E1}$
Robot-8-88-1-10	$2.01E3_{7.0E1}$▲	$1.93E3_{6.7E1}$	Robot-32-88-1-10	$1.39E3_{6.0E1}$−	$1.38E3_{4.5E1}$
Robot-16-88-0.1-0.1	$6.78E2_{3.5E0}$−	$6.78E2_{1.3E0}$	Robot-64-88-0.1-0.1	$2.13E2_{5.1E-1}$−	$2.13E2_{1.0E0}$
Robot-16-88-0.1-0.5	$1.91E3_{1.3E1}$▽	$1.92E3_{1.6E1}$	Robot-64-88-0.1-0.5	$8.75E2_{1.2E1}$▽	$8.78E2_{1.6E1}$
Robot-16-88-0.1-1	$1.15E3_{2.2E1}$−	$1.14E3_{2.4E1}$	Robot-64-88-0.1-1	$9.45E2_{1.2E1}$−	$9.46E2_{1.2E1}$
Robot-16-88-0.1-5	$5.16E2_{1.8E1}$▽	$5.19E2_{1.5E1}$	Robot-64-88-0.1-5	$2.55E3_{8.3E1}$▲	$2.50E3_{8.0E1}$
Robot-16-88-0.1-10	$2.53E3_{7.1E1}$▲	$2.46E3_{6.8E1}$	Robot-64-88-0.1-10	$3.27E3_{1.2E2}$▲	$3.17E3_{9.8E1}$
Robot-16-88-0.25-0.1	$6.00E1_{5.0E-1}$▽	$6.01E1_{5.3E-1}$	Robot-64-88-0.25-0.1	$1.35E3_{3.7E0}$−	$1.35E3_{5.4E-1}$
Robot-16-88-0.25-0.5	$3.67E2_{2.4E0}$−	$3.67E2_{2.1E0}$	Robot-64-88-0.25-0.5	$1.64E3_{1.7E1}$−	$1.64E3_{1.1E1}$
Robot-16-88-0.25-1	$5.48E2_{1.1E1}$−	$5.47E2_{8.2E0}$	Robot-64-88-0.25-1	$1.09E3_{1.0E1}$−	$1.09E3_{9.7E0}$
Robot-16-88-0.25-5	$3.78E3_{1.4E2}$−	$3.79E3_{1.2E2}$	Robot-64-88-0.25-5	$2.23E3_{7.3E1}$−	$2.22E3_{7.6E1}$
Robot-16-88-0.25-10	$3.53E3_{1.4E2}$▲	$3.46E3_{1.9E2}$	Robot-64-88-0.25-10	$1.72E2_{6.4E0}$▲	$1.69E2_{3.9E0}$
Robot-16-88-0.5-0.1	$9.34E2_{1.9E0}$▽	$9.35E2_{1.2E0}$	Robot-64-88-0.5-0.1	$3.57E2_{0.0E0}$−	$3.57E2_{0.0E0}$
Robot-16-88-0.5-0.5	$1.38E3_{2.0E1}$−	$1.37E3_{1.7E1}$	Robot-64-88-0.5-0.5	$1.10E3_{1.3E1}$−	$1.10E3_{1.3E1}$
Robot-16-88-0.5-1	$2.34E2_{2.8E0}$▽	$2.35E2_{2.7E0}$	Robot-64-88-0.5-1	$4.51E2_{4.0E0}$▲	$4.49E2_{5.6E0}$
Robot-16-88-0.5-5	$1.89E3_{7.5E1}$−	$1.89E3_{6.5E1}$	Robot-64-88-0.5-5	$1.79E3_{8.2E1}$−	$1.79E3_{7.7E1}$
Robot-16-88-0.5-10	$3.33E3_{1.1E2}$▲	$3.24E3_{1.1E2}$	Robot-64-88-0.5-10	$3.18E3_{1.1E2}$▲	$3.13E3_{9.4E1}$
Robot-16-88-0.75-0.1	$3.48E2_{3.6E-1}$−	$3.48E2_{7.4E-1}$	Robot-64-88-0.75-0.1	$8.15E2_{0.0E0}$−	$8.15E2_{0.0E0}$
Robot-16-88-0.75-0.5	$1.63E2_{1.6E0}$▲	$1.63E2_{1.6E0}$	Robot-64-88-0.75-0.5	$1.58E3_{1.3E1}$▲	$1.58E3_{5.7E0}$
Robot-16-88-0.75-1	$7.18E2_{2.0E1}$▲	$7.14E2_{1.9E1}$	Robot-64-88-0.75-1	$2.24E2_{5.0E0}$▲	$2.19E2_{5.5E0}$
Robot-16-88-0.75-5	$2.36E2_{7.6E0}$▲	$2.28E2_{6.9E0}$	Robot-64-88-0.75-5	$2.53E3_{9.0E1}$▲	$2.47E3_{6.5E1}$
Robot-16-88-0.75-10	$8.92E2_{4.6E1}$−	$8.98E2_{4.6E1}$	Robot-64-88-0.75-10	$4.92E3_{1.6E2}$▲	$4.86E3_{1.3E2}$
Robot-16-88-1-0.1	$2.81E2_{1.3E0}$−	$2.80E2_{1.0E0}$	Robot-64-88-1-0.1	$8.45E2_{0.0E0}$−	$8.45E2_{0.0E0}$
Robot-16-88-1-0.5	$5.14E2_{2.6E0}$▽	$5.15E2_{3.0E0}$	Robot-64-88-1-0.5	$9.58E2_{9.6E-1}$▽	$9.59E2_{3.1E0}$
Robot-16-88-1-1	$1.77E3_{3.6E1}$−	$1.77E3_{2.4E1}$	Robot-64-88-1-1	$2.47E2_{1.2E0}$▲	$2.45E2_{2.7E0}$
Robot-16-88-1-5	$1.15E3_{4.9E1}$−	$1.15E3_{4.5E1}$	Robot-64-88-1-5	$1.73E3_{4.1E1}$▲	$1.71E3_{5.6E1}$
Robot-16-88-1-10	$8.29E2_{2.9E1}$▲	$8.22E2_{2.9E1}$	Robot-64-88-1-10	$3.07E3_{1.3E2}$▲	$3.00E3_{1.1E2}$

Table 7. Sparse instances median and IQR of EFT-ILS and GRASP-CPA over 100 independent runs. Light gray emphasizes the best results.

Problem	EFT-ILS	GRASP-CPA	Problem	EFT-ILS	GRASP-CPA
Sparse-8-96-0.1-0.1	$2.63E1_{6.9E-1}-$	$2.63E1_{7.3E-1}$	Sparse-32-96-0.1-0.1	$3.70E2_{2.1E-1}-$	$3.70E2_{1.4E-1}$
Sparse-8-96-0.1-0.5	$2.68E2_{9.8E0}\blacktriangle$	$2.72E2_{1.1E1}$	Sparse-32-96-0.1-0.5	$4.96E2_{1.5E1}-$	$4.94E2_{1.3E1}$
Sparse-8-96-0.1-1	$4.50E2_{2.0E1}-$	$4.56E2_{2.0E1}$	Sparse-32-96-0.1-1	$5.33E2_{2.2E1}\triangledown$	$5.34E2_{1.2E1}$
Sparse-8-96-0.1-5	$7.65E2_{2.6E1}\blacktriangle$	$7.56E2_{3.2E1}$	Sparse-32-96-0.1-5	$2.36E2_{3.4E0}-$	$2.36E2_{4.8E0}$
Sparse-8-96-0.1-10	$8.93E2_{4.0E1}-$	$8.89E2_{4.0E1}$	Sparse-32-96-0.1-10	$1.41E3_{5.8E0}\blacktriangle$	$1.41E3_{7.8E0}$
Sparse-8-96-0.25-0.1	$5.10E2_{1.1E1}-$	$5.08E2_{1.3E1}$	Sparse-32-96-0.25-0.1	$1.95E2_{8.5E0}-$	$1.96E2_{7.6E0}$
Sparse-8-96-0.25-0.5	$2.20E1_{7.7E-1}-$	$2.21E1_{8.0E-1}$	Sparse-32-96-0.25-0.5	$1.41E2_{2.4E0}-$	$1.41E2_{4.2E0}$
Sparse-8-96-0.25-1	$3.28E2_{1.9E1}\triangledown$	$3.30E2_{1.5E1}$	Sparse-32-96-0.25-1	$4.32E2_{1.2E1}-$	$4.33E2_{1.3E1}$
Sparse-8-96-0.25-5	$2.09E2_{9.2E0}\blacktriangle$	$2.07E2_{9.8E0}$	Sparse-32-96-0.25-5	$6.10E2_{4.1E1}-$	$6.14E2_{4.6E1}$
Sparse-8-96-0.25-10	$1.46E3_{5.6E1}-$	$1.45E3_{4.6E1}$	Sparse-32-96-0.25-10	$7.48E2_{9.5E0}-$	$7.48E2_{7.2E0}$
Sparse-8-96-0.5-0.1	$8.45E2_{3.1E1}-$	$8.55E2_{3.5E1}$	Sparse-32-96-0.5-0.1	$3.68E2_{1.6E1}-$	$3.66E2_{1.3E1}$
Sparse-8-96-0.5-0.5	$8.47E2_{2.1E1}-$	$8.44E2_{3.0E1}$	Sparse-32-96-0.5-0.5	$3.47E1_{1.2E0}\blacktriangle$	$3.42E1_{1.1E0}$
Sparse-8-96-0.5-1	$5.95E2_{2.0E1}-$	$5.98E2_{2.6E1}$	Sparse-32-96-0.5-1	$5.27E2_{1.7E1}-$	$5.23E2_{1.8E1}$
Sparse-8-96-0.5-5	$3.56E1_{1.6E0}\blacktriangle$	$3.52E1_{1.2E0}$	Sparse-32-96-0.5-5	$3.54E2_{1.8E1}-$	$3.54E2_{1.2E1}$
Sparse-8-96-0.5-10	$5.73E2_{3.2E1}-$	$5.69E2_{2.4E1}$	Sparse-32-96-0.5-10	$4.28E2_{2.6E0}\blacktriangle$	$4.27E2_{3.2E0}$
Sparse-8-96-0.75-0.1	$6.50E2_{1.3E1}\blacktriangle$	$6.43E2_{1.7E1}$	Sparse-32-96-0.75-0.1	$1.20E2_{4.9E0}-$	$1.21E2_{4.9E0}$
Sparse-8-96-0.75-0.5	$2.67E2_{7.0E0}\blacktriangle$	$2.64E2_{7.9E0}$	Sparse-32-96-0.75-0.5	$2.15E2_{5.0E0}-$	$2.15E2_{6.0E0}$
Sparse-8-96-0.75-1	$6.52E2_{2.3E1}\blacktriangle$	$6.46E2_{2.1E1}$	Sparse-32-96-0.75-1	$2.25E2_{8.9E0}-$	$2.28E2_{6.6E0}$
Sparse-8-96-0.75-5	$9.51E2_{8.3E1}-$	$9.51E2_{8.1E1}$	Sparse-32-96-0.75-5	$3.59E1_{1.1E0}\blacktriangle$	$3.58E1_{1.0E0}$
Sparse-8-96-0.75-10	$9.99E2_{7.1E1}\blacktriangle$	$9.36E2_{6.0E1}$	Sparse-32-96-0.75-10	$6.68E2_{2.2E1}\blacktriangle$	$6.61E2_{2.9E1}$
Sparse-8-96-1-0.1	$6.97E2_{1.6E1}\blacktriangle$	$6.92E2_{1.5E1}$	Sparse-32-96-1-0.1	$2.98E1_{5.7E-1}-$	$2.99E1_{5.7E-1}$
Sparse-8-96-1-0.5	$6.47E2_{2.4E1}\blacktriangle$	$6.30E2_{2.5E1}$	Sparse-32-96-1-0.5	$2.88E2_{7.3E0}-$	$2.88E2_{7.2E0}$
Sparse-8-96-1-1	$2.48E2_{1.1E1}-$	$2.47E2_{7.0E0}$	Sparse-32-96-1-1	$2.05E2_{7.2E0}\triangledown$	$2.08E2_{6.4E0}$
Sparse-8-96-1-5	$5.41E2_{2.3E1}\blacktriangle$	$5.34E2_{2.2E1}$	Sparse-32-96-1-5	$5.95E2_{3.2E1}\blacktriangle$	$5.86E2_{1.9E1}$
Sparse-8-96-1-10	$1.96E2_{9.8E0}-$	$1.97E2_{1.3E1}$	Sparse-32-96-1-10	$1.01E3_{7.2E1}\blacktriangle$	$9.81E2_{9.4E1}$
Sparse-16-96-0.1-0.1	$5.47E2_{1.5E1}-$	$5.46E2_{2.1E1}$	Sparse-64-96-0.1-0.1	$1.36E2_{4.2E-1}\triangledown$	$1.37E2_{3.7E-2}$
Sparse-16-96-0.1-0.5	$3.55E2_{1.3E1}-$	$3.57E2_{1.3E1}$	Sparse-64-96-0.1-0.5	$4.74E2_{9.6E-1}\triangledown$	$4.74E2_{2.7E0}$
Sparse-16-96-0.1-1	$4.59E2_{2.0E1}-$	$4.60E2_{1.3E1}$	Sparse-64-96-0.1-1	$3.12E2_{1.6E0}\triangledown$	$3.15E2_{2.2E0}$
Sparse-16-96-0.1-5	$1.91E2_{5.7E0}\blacktriangle$	$1.89E2_{5.8E0}$	Sparse-64-96-0.1-5	$3.30E2_{9.9E-1}\blacktriangle$	$3.29E2_{1.2E0}$
Sparse-16-96-0.1-10	$5.54E2_{7.9E0}-$	$5.54E2_{8.4E0}$	Sparse-64-96-0.1-10	$1.39E2_{4.0E-1}\blacktriangle$	$1.38E2_{2.2E-1}$
Sparse-16-96-0.25-0.1	$3.04E2_{1.1E1}-$	$3.05E2_{1.1E1}$	Sparse-64-96-0.25-0.1	$2.73E2_{2.2E-1}\blacktriangle$	$2.73E2_{0.0E0}$
Sparse-16-96-0.25-0.5	$2.27E2_{9.6E0}-$	$2.28E2_{8.3E0}$	Sparse-64-96-0.25-0.5	$4.04E2_{4.6E0}\triangledown$	$4.05E2_{1.1E0}$
Sparse-16-96-0.25-1	$2.75E2_{1.2E1}\triangledown$	$2.78E2_{1.2E1}$	Sparse-64-96-0.25-1	$9.32E1_{2.5E-1}-$	$9.32E1_{9.5E-2}$
Sparse-16-96-0.25-5	$8.10E2_{3.4E1}-$	$8.07E2_{3.2E1}$	Sparse-64-96-0.25-5	$3.86E2_{3.0E1}\blacktriangle$	$3.64E2_{1.9E1}$
Sparse-16-96-0.25-10	$1.09E3_{4.5E1}\blacktriangle$	$1.06E3_{3.2E1}$	Sparse-64-96-0.25-10	$5.47E2_{4.6E-1}-$	$5.47E2_{4.6E-1}$
Sparse-16-96-0.5-0.1	$1.92E2_{8.4E0}-$	$1.94E2_{5.8E0}$	Sparse-64-96-0.5-0.1	$4.52E2_{2.0E0}-$	$4.52E2_{5.8E-1}$
Sparse-16-96-0.5-0.5	$2.80E2_{1.5E1}-$	$2.82E2_{8.6E0}$	Sparse-64-96-0.5-0.5	$1.26E2_{7.9E0}\blacktriangle$	$1.21E2_{6.7E0}$
Sparse-16-96-0.5-1	$2.01E2_{9.3E0}-$	$2.04E2_{9.0E0}$	Sparse-64-96-0.5-1	$2.73E2_{3.4E0}\triangledown$	$2.75E2_{2.5E0}$
Sparse-16-96-0.5-5	$7.70E2_{5.3E1}\blacktriangle$	$7.59E2_{3.9E1}$	Sparse-64-96-0.5-5	$7.03E2_{1.0E1}-$	$7.00E2_{2.0E1}$
Sparse-16-96-0.5-10	$1.80E3_{2.3E1}-$	$1.80E3_{3.3E1}$	Sparse-64-96-0.5-10	$1.74E3_{0.0E0}\blacktriangle$	$1.74E3_{0.0E0}$
Sparse-16-96-0.75-0.1	$3.53E2_{1.5E1}-$	$3.51E2_{1.5E1}$	Sparse-64-96-0.75-0.1	$2.89E2_{5.2E0}-$	$2.91E2_{5.2E0}$
Sparse-16-96-0.75-0.5	$2.93E2_{1.2E1}-$	$2.93E2_{9.4E0}$	Sparse-64-96-0.75-0.5	$3.57E2_{1.1E0}\blacktriangle$	$3.57E2_{5.3E0}$
Sparse-16-96-0.75-1	$3.46E2_{1.5E1}\blacktriangle$	$3.42E2_{1.4E1}$	Sparse-64-96-0.75-1	$3.68E2_{3.4E0}\triangledown$	$3.72E2_{1.2E1}$
Sparse-16-96-0.75-5	$5.94E2_{1.7E1}-$	$5.95E2_{2.4E1}$	Sparse-64-96-0.75-5	$3.13E2_{2.2E0}-$	$3.13E2_{0.0E0}$
Sparse-16-96-0.75-10	$1.64E3_{3.8E1}-$	$1.63E3_{6.6E1}$	Sparse-64-96-0.75-10	$4.64E2_{3.3E0}-$	$4.64E2_{3.0E0}$
Sparse-16-96-1-0.1	$2.03E2_{7.2E0}-$	$2.01E2_{7.2E0}$	Sparse-64-96-1-0.1	$2.32E2_{0.0E0}-$	$2.32E2_{0.0E0}$
Sparse-16-96-1-0.5	$2.49E2_{1.2E1}-$	$2.50E2_{1.2E1}$	Sparse-64-96-1-0.5	$1.72E2_{4.4E0}-$	$1.72E2_{2.5E0}$
Sparse-16-96-1-1	$1.79E2_{8.0E0}-$	$1.78E2_{7.5E0}$	Sparse-64-96-1-1	$8.33E1_{0.0E0}\blacktriangle$	$8.33E1_{0.0E0}$
Sparse-16-96-1-5	$3.73E2_{1.9E1}-$	$3.73E2_{2.0E1}$	Sparse-64-96-1-5	$4.10E2_{9.4E0}\blacktriangle$	$4.07E2_{8.6E0}$
Sparse-16-96-1-10	$1.75E3_{5.2E1}\blacktriangle$	$1.73E3_{4.9E1}$	Sparse-64-96-1-10	$1.65E3_{1.1E0}\blacktriangle$	$1.65E3_{0.0E0}$

Table 8. Small synthetic benchmark instances median and IQR of EFT-ILS and GRASP-CPA over 100 independent runs. Light gray emphasizes the best results.

Problem	EFT-ILS	GRASP-CPA	Optimum	Reference	$\text{Time}_{\text{Enum}}$	$\text{Time}_{\text{EFT-ILS}}$	$\text{Time}_{\text{GRASP-CPA}}$
Ahmad-3-9	$27_{0.0}-$	$27_{0.0}$	27	[42]	8s	0s	0s
Daoud-2-11	$60_{1.5}\blacktriangle$	$58.5_{4.0}$	56	[43]	46s	0s	0s
Eswari-2-11	$60_{0.5}\blacktriangle$	$58.5_{4.0}$	56	[44]	46s	0s	0s
Hamid-3-10	$100_{0.0}-$	$100_{0.0}$	100	[45]	71s	0s	0s
Heteropar-4-12	$136_{0.0}\blacktriangle$	124_{12}	124	[46]	1677s	0s	0s
Hsu-3-10	$80_{0.0}\blacktriangle$	$80_{0.0}$	80	[47]	64s	0s	0s
Ilavarasan-3-10	$73_{3.0}\blacktriangle$	$73_{0.0}$	73	[48]	65s	0s	0s
Kang1-3-10	$76_{0.0}\blacktriangle$	$76_{1.0}$	73	[49]	67s	0s	0s
Kang2-3-10	$83_{0.0}\blacktriangle$	$83_{0.0}$	79	[50]	68s	0s	0s
Kuan-3-10	$27_{0.0}\blacktriangle$	$27_{0.0}$	26	[51]	25s	0s	0s
Liang-3-10	$73_{3.0}\blacktriangle$	$73_{3.0}$	73	[52]	67s	0s	0s
Sample-3-8	$84_{2.0}\blacktriangle$	$82_{3.0}$	81	[11]	0s	0s	0s
SampleFig-3-8	$66_{0.0}-$	$66_{0.0}$	66	[11]	0s	0s	0s
YCLee-3-8	$66_{0.0}-$	$66_{0.0}$	66	[53]	0s	0s	0s

Table 9. Small synthetic benchmark instances best results found by EFT-ILS and GRASP-CPA over 100 independent runs. Light gray emphasizes the best results.

Problem	EFT-ILS	GRASP-CPA	Optimum	Reference
Ahmad-3-9	27	27	27	[42]
Daoud-2-11	56	56	56	[43]
Eswari-2-11	56	56	56	[44]
Hamid-3-10	100	100	100	[45]
Heteropar-4-12	136	124	124	[46]
Hsu-3-10	80	80	80	[47]
Ilavarasan-3-10	73	73	73	[48]
Kang1-3-10	73	73	73	[49]
Kang2-3-10	83	82	79	[50]
Kuan-3-10	26	26	26	[51]
Liang-3-10	73	73	73	[52]
Sample-3-8	81	81	81	[11]
SampleFig-3-8	66	66	66	[11]
YCLee-3-8	66	66	66	[53]

6. Conclusions and Future Work

This paper proposes a new Cellular Processing Algorithm that uses a GRASP construction, called GRASP-CPA, for scheduling precedence-constraint tasks on heterogeneous systems. Experimental results showed that GRASP-CPA outperformed a high-performance algorithm from the state-of-the-art called EFT-ILS, regarding optimal and median values with statistical significance for the proposed set of instances. Two main features of the GRASP-CPA contribute to its performance. The first is the generation of tasks' execution orders using a GRASP algorithm, conversely to a completely random order generation in EFT-ILS. The second is the communication between different Processing Cells to help explore the search space. We encourage researches to apply the Cellular Processing Algorithm approach to their problems. This approach is more a framework than a strict algorithm that allows flexible implementations with homogeneous or heterogeneous Processing Cells. Despite being a novel approach, it has proven to be effective for several problems and still has many open research areas. As future work, we would like to research other methods to produce tasks' execution orderings, to improve the results yielded by GRASP-CPA.

Author Contributions: Conceptualization, formal analysis, investigation, methodology, resources, software, supervision, validation, writing—original draft, writing—review and editing, A.S.; data curation, project administration, software, validation, writing—original draft, and writing—review and editing, J.D.T.-V.; funding acquisition, project administration, visualization, S.I.M.; writing—review and editing, J.A.C.R., J.L.M., M.G.T.B., and M.P.-F. All authors have read and agreed to the published version of the manuscript.

Funding: A. Santiago would like to thank the CONACyT Mexico SNI for the salary award under the record 83525. The APC was funded by the Universidad Autónoma de Tamaulipas, grant PROFEXCE 2020.

Acknowledgments: We would like to thank Johnatan E. Pecero for helping with the scheduling problem instances for the experimental comparison.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Soto, C.; Santiago, A.; Fraire, H.; Dorronsoro, B. Optimal Scheduling for Precedence-Constrained Applications on Heterogeneous Machines. *Int. Conf. Ser. Multidiscipl. Sci.* **2018**. [CrossRef]
2. Ullman, J. NP-complete scheduling problems. *J. Comput. Syst. Sci.* **1975**, *10*, 384–393. [CrossRef]
3. Sinnen, O. *Task Scheduling for Parallel Systems*; John Wiley & Sons: Hoboken, NJ, USA, 2007; Volume 60.
4. Topcuoglu, H.; Hariri, S.; Wu, M.-Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* **2002**, *13*, 260–274. [CrossRef]
5. Diaz, C.O.; Guzek, M.; Pecero, J.E.; Danoy, G.; Bouvry, P.; Khan, S.U. Energy-aware fast scheduling heuristics in heterogeneous computing systems. In Proceedings of the 2011 International Conference on High Performance Computing Simulation, Instanbul, Turkey, 4–8 July 2011; pp. 478–484.
6. Diaz, C.O.; Pecero, J.E.; Bouvry, P. Scalable, low complexity, and fast greedy scheduling heuristics for highly heterogeneous distributed computing systems. *J. Supercomput.* **2014**, *67*, 837–853. [CrossRef]
7. Lee, Y.C.; Zomaya, A.Y. Minimizing Energy Consumption for Precedence-Constrained Applications Using Dynamic Voltage Scaling. In Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai, China, 18–21 May 2009; pp. 92–99.
8. Pecero, J.E.; Huacuja, H.J.F.; Bouvry, P.; Pineda, A.A.S.; Locés, M.C.L.; Barbosa, J.J.G. On the energy optimization for precedence constrained applications using local search algorithms. In Proceedings of the 2012 International Conference on High Performance Computing Simulation (HPCS), Madrid, Spain, 2–6 July 2012; pp. 133–139. [CrossRef]
9. Pineda, A.A.S. Estrategias de Búsqueda local Para el Problema de Progamación de Tareas en Sistemas de Procesamiento Paralelo. Master's Thesis, Instituto Tecnológico de Ciudad Madero, Cd Madero, Mexico, 2013.
10. Nesmachnow, S.; Luna, F.; Alba, E. An Efficient Stochastic Local Search for Heterogeneous Computing Scheduling. In Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum, Shanghai, China, 21–25 May 2012; pp. 593–600. [CrossRef]
11. Pineda, A.A.S.; Pecero, J.; Huacuja, H.; Barbosa, J.; Bouvry, P. An iterative local search algorithm for scheduling precedence-constrained applications on heterogeneous machines. In Proceedings of the 6th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2013), Ghent, Belgium, 27–29 August 2013; pp. 472–485.
12. Huacuja, H.J.F.; Santiago, A.; Pecero, J.E.; Dorronsoro, B.; Bouvry, P.; Monterrubio, J.C.S.; Barbosa, J.J.G.; Santillan, C.G. A Comparison Between Memetic Algorithm and Seeded Genetic Algorithm for Multi-Objective Independent Task Scheduling on Heterogeneous Machines. In *Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization*; Melin, P., Castillo, O., Kacprzyk, J., Eds.; Springer International Publishing: Cham, Switzerlands, 2015; pp. 377–389. [CrossRef]
13. Pecero, J.E.; Bouvry, P.; Huacuja, H.J.F.; Villanueva, J.D.T.; Zuñiga, M.A.R.; Santillán, C.G.G. Task Scheduling in Heterogeneous Computing Systems Using a MicroGA. In Proceedings of the 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Compiegne, France, 28–30 October 2013; pp. 618–623.
14. Flórez, E.; Barrios, C.J.; Pecero, J.E. Methods for Job Scheduling on Computational Grids: Review and Comparison. In *High Performance Computing*; Osthoff, C., Navaux, P.O.A., Barrios Hernandez, C.J., Silva Dias, P.L., Eds.; Springer International Publishing: Cham, Switzerlands, 2015; pp. 19–33.

15. Pecero, J.E.; Bouvry, P.; Huacuja, H.J.F.; Khan, S.U. A Multi-objective GRASP Algorithm for Joint Optimization of Energy Consumption and Schedule Length of Precedence-Constrained Applications. In Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, Sydney, Australia, 12–14 December 2011; pp. 510–517.
16. Huang, Q.; Su, S.; Li, J.; Xu, P.; Shuang, K.; Huang, X. Enhanced Energy-Efficient Scheduling for Parallel Applications in Cloud. In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), Ottawa, ON, Canada, 13–16 May 2012; pp. 781–786.
17. Lee, Y.C.; Zomaya, A.Y. Energy Conscious Scheduling for Distributed Computing Systems under Different Operating Conditions. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *22*, 1374–1381. [CrossRef]
18. Xiao, X.; Xie, G.; Xu, C.; Fan, C.; Li, R.; Li, K. Maximizing reliability of energy constrained parallel applications on heterogeneous distributed systems. *J. Comput. Sci.* **2018**, *26*, 344–353. [CrossRef]
19. Ait Aba, M.; Zaourar, L.; Munier, A. Efficient Algorithm for Scheduling Parallel Applications on Hybrid Multicore Machines with Communications Delays and Energy Constraint. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5573. [CrossRef]
20. Chen, W.; Xie, G.; Li, R.; Bai, Y.; Fan, C.; Li, K. Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems. *Future Gener. Comput. Syst.* **2017**, *74*, 1–11. [CrossRef]
21. Xie, G.; Chen, Y.; Xiao, X.; Xu, C.; Li, R.; Li, K. Energy-Efficient Fault-Tolerant Scheduling of Reliable Parallel Applications on Heterogeneous Distributed Embedded Systems. *IEEE Trans. Sustain. Comput.* **2018**, *3*, 167–181. [CrossRef]
22. Xiaoyong, T.; Li, K.; Zeng, Z.; Veeravalli, B. A Novel Security-Driven Scheduling Algorithm for Precedence-Constrained Tasks in Heterogeneous Distributed Systems. *IEEE Trans. Comput.* **2011**, *60*, 1017–1029. [CrossRef]
23. Lee, Y.C.; Zomaya, A.Y.; Siegel, H.J. Robust task scheduling for volunteer computing systems. *J. Supercomput.* **2010**, *53*, 163–181. [CrossRef]
24. Applegate, D.; Cook, W. A Computational Study of the Job-Shop Scheduling Problem. *ORSA J. Comput.* **1991**, *3*, 149–156. [CrossRef]
25. Soto, C.; Dorronsoro, B.; Fraire, H.; Cruz-Reyes, L.; Gomez-Santillan, C.; Rangel, N. Solving the multi-objective flexible job shop scheduling problem with a novel parallel branch and bound algorithm. *Swarm Evol. Comput.* **2020**, *53*, 100632. [CrossRef]
26. Terán-Villanueva, J.D.; Fraire-Huacuja, H.J.; Carpio-Valadez, J.M.; Pazos R., R.A.; Puga-Soberanes, H.J.; Martínez-Flores, J.A. Experimental study of a new algorithm-design-framework based on cellular computing. In *Studies in Computational Intelligence*; Castillo, O., Melin, P., Kacprzyk, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 451, pp. 517–532. [CrossRef]
27. Terán-Villanueva, J.D.; Fraire-Huacuja, H.J.; Carpio Valadez, J.M.; Pazos R., R.A.; Puga-Soberanes, H.J.; Martínez-Flores, J.A. Cellular processing algorithms. In *Studies in Fuzziness and Soft Computing*; Melin, P., Castillo, O., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 294, pp. 53–74. [CrossRef]
28. Terán-Villanueva, J.D.; Fraire-Huacuja, H.J.; Ibarra Martínez, S.; Cruz-Reyes, L.; Castán Rocha, J.A.; Gómez Santillán, C.; Menchaca, J.L. Cellular processing algorithm for the vertex bisection problem: Detailed analysis and new component design. *Inf. Sci.* **2019**, *478*, 62–82. [CrossRef]
29. Terán-Villanueva, J.D.; Fraire-Huacuja, H.J.; Carpio-Valadez, J.M.; Pazos R., R.A.; Puga-Soberanes, H.J.; Martínez-Flores, J.A. *Cellular Processing Scatter Search for Minimizing Power Consumption on Wireless Communications Systems*; Hpcs: Madrid, España, 2012; pp. 126–132.
30. Terán-Villanueva, J.D.; Fraire Huacuja, H.J.; Carpio Valadez, J.M.; Pazos Rangel, R.; Puga Soberanes, H.J.; Martínez Flores, J.A. A heterogeneous cellular processing algorithm for minimizing the power consumption in wireless communications systems. *Comput. Optim. Appl.* **2015**, *62*, 787–814. [CrossRef]
31. Lourenço, H.R.; Martin, O.C.; Stützle, T. Iterated Local Search: Framework and Applications. In *Handbook of Metaheuristics*; Gendreau, M., Potvin, J.Y., Eds.; Springer: Boston, MA, USA, 2010; pp. 363–397. [CrossRef]
32. Hoos, H.H.; Stützle, T. *Stochastic Local Search: Foundations and Applications*; Elsevier: Amsterdam, The Netherlands, 2004.

33. Resende, M.G.; Ribeiro, C.C. Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications. In *Handbook of Metaheuristics*; Gendreau, M., Potvin, J.Y., Eds.; Springer: Boston, MA, USA, 2010; pp. 283–319. [CrossRef]
34. Monterrubio, J.C.S.; Huacuja, H.J.F.; Alejandro, A.; Pineda, S. *Comparativa de tres Cruzas y Cuatro Mutaciones Para el Problema de Asignación de Tareas en Sistemas de Cómputo Heterogéneo*; Instituto Tecnológico de Ciudad Madero: Ciudad Madero, Mexico, 2015. [CrossRef]
35. Saavedra, R.H.; Smith, A.J. Analysis of benchmark characteristics and benchmark performance prediction. *ACM Trans. Comput. Syst. (TOCS)* **1996**, *14*, 344–384. [CrossRef]
36. Brown, D.A.; Brady, P.R.; Dietz, A.; Cao, J.; Johnson, B.; McNabb, J. A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis. In *Workflows for e-Science: Scientific Workflows for Grids*; Springer: London, UK, 2007; Chapter 4, pp. 39–59. [CrossRef]
37. Tobita, T.; Kasahara, H. A standard task graph set for fair evaluation of multiprocessor scheduling algorithms. *J. Sched.* **2002**, *5*, 379–394. [CrossRef]
38. Santiago, A.; Terán-Villanueva, J.D.; Martínez, S.I.; Rocha, J.A.C.; Menchaca, J.L.; Borreones, M.G.T.; Ponce-Flores, M. Instance Set for: GRASP and Iterated Local Search Based Cellular Processing Algorithm for Precedence-Constraint Task List Scheduling on Heterogeneous Systems. Available online: https://github.com/AASantiago/SchedulingInstances (accessed on 27 August 2020).
39. Santiago, A.; Terán-Villanueva, J.D.; Martínez, S.I.; Rocha, J.A.C.; Menchaca, J.L.; Borreones, M.G.T.; Ponce-Flores, M. Source Code for: GRASP and Iterated Local Search Based Cellular Processing Algorithm for Precedence-Constraint Task List Scheduling on Heterogeneous Systems. 2020. Available online: https://github.com/AASantiago/GRASP-CPA (accessed on 27 August 2020).
40. Pineda, A.A.S.; Pecero, J.E.; Huacuja, H.J.F.; Barbosa, J.J.G.; Bouvry, P. Source Code for: An Iterative Local Search Algorithm for Scheduling Precedence-Constrained Applications on Heterogeneous Machines. 2020. Available online: https://github.com/AASantiago/EFT-ILS (accessed on 27 August 2020).
41. Corder, G.W.; Foreman, D.I. *Nonparametric Statistics for Non-Statisticians*; Wiley: Hoboken, NJ, USA, 2011.
42. Ahmad, I.; Dhodhi, M.K.; Ul-Mustafa, R. DPS: Dynamic priority scheduling heuristic for heterogeneous computing systems. *IEE Proc. Comput. Digit. Tech.* **1998**, *145*, 411–418. [CrossRef]
43. Daoud, M.I.; Kharma, N. A hybrid heuristic–genetic algorithm for task scheduling in heterogeneous processor networks. *J. Parallel Distrib. Comput.* **2011**, *71*, 1518–1531. [CrossRef]
44. Eswari, R.; Nickolas, S. Path-Based Heuristic Task Scheduling Algorithm for Heterogeneous Distributed Computing Systems. In Proceedings of the 2010 International Conference on Advances in Recent Technologies in Communication and Computing, Kottayam, India, 16–17 October 2010; pp. 30–34.
45. Arabnejad, H. List Based Task Scheduling Algorithms on Heterogeneous Systems—An Overview. Available online: https://paginas.fe.up.pt/~prodei/dsie12/papers/paper_30.pdf (accessed on 27 August 2020).
46. Arabnejad, H.; Barbosa, J.G. Performance Evaluation of List Based Scheduling on Heterogeneous Systems. Available online: http://icl.cs.utk.edu/workshops/heteropar2011/slides/heteropar_JorgeBarbosa.pdf (accessed on 27 August 2020).
47. Hsu, C.H.; Hsieh, C.W.; Yang, C.T. A generalized critical task anticipation technique for dag scheduling. In *International Conference on Algorithms and Architectures for Parallel Processing*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 493–505.
48. Ilavarasan, E.; Thambidurai, P.; Mahilmannan, R. Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System. In Proceedings of the 4th International Symposium on Parallel and Distributed Computing (ISPDC'05), Lile, France, 4–6 July 2005; pp. 28–38.
49. Kang, Y.; Zhang, Z.; Chen, P. An activity-based genetic algorithm approach to multiprocessor scheduling. In Proceedings of the 2011 Seventh International Conference on Natural Computation, Shanghai, China, 26–28 July 2011; Volume 2, pp. 1048–1052.
50. Kang, Y.; Lin, Y. A recursive algorithm for scheduling of tasks in a heterogeneous distributed environment. In Proceedings of the 2011 4th International Conference on Biomedical Engineering and Informatics (BMEI), Shanghai, China, 15–17 October 2011; Volume 4, pp. 2099–2103.
51. Lai, K.C.; Yang, C.T. A dominant predecessor duplication scheduling algorithm for heterogeneous systems. *J. Supercomput.* **2008**, *44*, 126–145. [CrossRef]

52. Lee, L.; Chen, C.; Chang, H.; Tang, C.; Pan, K. A Non-Critical Path Earliest-Finish Algorithm for Inter-dependent Tasks in Heterogeneous Computing Environments. In Proceedings of the 2009 11th IEEE International Conference on High Performance Computing and Communications, Seoul, Korea, 25–27 June 2009; pp. 603–608.
53. Lee, Y.; Zomaya, A. A Novel State Transition Method for Metaheuristic-Based Scheduling in Heterogeneous Computing Systems. *IEEE Trans. Parallel Distrib. Syst.* **2008**, *19*, 1215–1223.

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).