Expert Systems with Applications 41 (2014) 2496–2513

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Solving system-level synthesis problem by a multi-objective estimation of distribution algorithm

Ling Wang [a,*], Chen Fang [a], Ponnuthurai Nagaratnam Suganthan [b], Min Liu [a]

[a] Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China
[b] School of Electrical and Electronic Engineering, Nanyang Technological University, 639798 Singapore, Singapore

ARTICLE INFO

Keywords:
Estimation of distribution algorithm
Probability model
System-level synthesis problem
Project scheduling
Image compression standard

ABSTRACT

In this paper, the system-level synthesis problem (SLSP) is modeled as a multi-objective mode-identity resource-constrained project scheduling problem with makespan and resource investment criteria (MOMIRCPSP-MS-RI). Then, a hybrid Pareto-archived estimation of distribution algorithm (HPAEDA) is presented to solve the MOMIRCPSP-MS-RI. To be specific, the individual of the population is encoded as the activity-mode-priority-resource list (AMPRL), and a hybrid probability model is used to predict the most promising search area, and a Pareto archive is used to preserve the non-dominated solutions that have been explored, and another archive is used to preserve the solutions for updating the probability model. Moreover, specific sampling mechanism and updating mechanism for the probability model are both provided to track the most promising search area via the EDA-based evolutionary search. Finally, the modeling methodology and the HPAEDA are tested by an example of a video codec based on the H.261 image compression standard. Simulation results and comparisons demonstrate the effectiveness of the modeling methodology and the proposed algorithm.

1. Introduction

With the development of VLSI (Very-large-scale integration) technology, semiconductor companies like Intel can build large-scale, complex electronic systems which contain millions of transistors on a single chip. Meanwhile, due to the increasing system complexities, the system-level synthesis problem (SLSP) has emerged (Gerstlauer et al., 2009). There is a need for moving to the system level of abstraction in order to increase productivity in electronic system design. Different from high-level synthesis which is devoted to mapping behavioral description to resistor transistor logic (RTL) (Rosado-Munoz, Bataller-Mompeán, Soria-Olivas, Scarante, & Guerrero-Martínez, 2011), system-level synthesis considers the system hardware and software design simultaneously. Recently, some tools have emerged to realize and support the system-level synthesis process, such as Daedalus (Nikolov, Stefanov, & Deprettere, 2008; Nikolov et al., 2008), SoC Environment (SCE) (Dömer et al., 2008), and SystemCoDesigner (Keinert et al., 2009).

To solve the SLSP, the mixed integer linear programming (MILP) is widely used (Schwiegershausen, Kropp, & Pirsch, 1996; Niemann & Marwedel, 1997; Nagaraj Shenoy, Banerjee, & Choudhary, 2000). However, it has some disadvantages in solving the SLSP with the MILP. First, the MILP can only solve the small-scaled problems (no more than 20 tasks) in a reasonable computation time since the SLSP is NP-hard in a general case (Mann & Orbán, 2003). Second, it is difficult for the MILP to solve the SLSP with multiple objectives. As a result, it usually adopts the MILP to solve one chosen objective, and then uses the high-level synthesis tools to solve other objectives. But the MILP-based procedure still cannot guarantee the Pareto optimal solutions.

Since many real world problems are difficult to solve by traditional methods, soft computing has gained much attention during recent years in many fields, such as controller design (Wang & Li, 2011), engineering design (Zhao & Wang, 2011; Zhao, Wang, Zeng, & Fan, 2012), steelmaking scheduling (Pan, Wang, Mao, Zhao, & Zhang, 2013), and economic load dispatch (Wang & Li, 2013). During the past few years, evolutionary algorithm (EA) has also been used to solve the SLSP. Blickle (1996) first developed a single-objective EA, and later Blickle, Teich, and Thiele (1998) introduced a Pareto-ranking technique into the single-objective EA. Fan, Wang, Achiche, Goodman, and Rosenberg (2008) introduced the flow of a structured Micro-Electro-Mechanical Systems (MEMS) design process to emphasize the system-level lumped-parameter model synthesis. To trade off the predefined behavioral specifications for designers, at the system level an approach combining bond graphs and genetic programming can yield satisfactory design candidates. Zitzler and Thiele (1999) developed a multi-objective algorithm named SPEA to solve the SLSP by combining several features of

* Corresponding author. Tel.: +86 10 62783125; fax: +86 10 62786911.
E-mail address: wangling@mail.tsinghua.edu.cn (L. Wang).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.09.049

previous multi-objective EAs in a unique manner. Compared to the MILP-based procedures, the EA-based procedures can deal with the large-scaled and multi-objective problems. However, the specific search operators should be designed for the EAs to solve the SLSP due to the complicated constraints. So, it is important to develop novel methodologies to model the problem reasonably as well as powerful solution algorithms to solve the problem effectively.

As a novel evolutionary algorithm, estimation of distribution algorithm (EDA) can be regarded as a general framework of statistical learning based optimization algorithm (Larrañaga & Lozano, 2002). Unlike genetic algorithm (GA) which explicitly generates new individuals by crossover and mutation, the EDA tries to predict of the movement of population in the search space and estimates the underlying probability distribution of the encoded variables of the elite individuals so as to generate new individuals. So far, the EDA has been applied to solve a variety of optimization problems in academic and industrial fields, such as feature selection (Armañanza et al., 2011), shop scheduling (Wang, Wang, Xu, Zhou, & Liu, 2012), nurse rostering (Aickelin, Burke, & Li, 2007), hybrid electric vehicle charging (Su & Chow, 2012), multi-speed planetary transmission (Simionescu, Beale, & Dozier, 2006), knapsack problem (Wang, Wang, & Xu, 2012), and software testing (Sagarna & Lozano, 2005). However, to the best of our knowledge, there is no work about EDA to solve the SLSP.

In this paper, the SLSP is solved by adopting the project scheduling concept-based model and using the EDA-based search method. First, the SLSP is modeled as a multi-objective mode-identity resource-constrained project scheduling problem with makespan and resource investment criteria (MOMIRCPSP-MS-RI). Then, a hybrid Pareto-archived estimation of distribution algorithm (HPAEDA) is proposed to solve the problem. The activity-mode-priority-resource list (AMPRL) is used to encode individuals, and a hybrid probability model is designed to predict the promising search area. During the search procedure, a Pareto archive is employed to preserve the non-dominated solutions that have been explored, and another archive is used to preserve the solutions for updating the probability model. Specific sampling and updating mechanisms are designed to make the evolution process track the most promising search areas. The modeling methodology and the proposed HPAEDA are tested with the example of a video codec based on the H.261 image compression standard. Simulation results and comparisons demonstrate the effectiveness of the modeling methodology and the proposed HPAEDA.

The remainder of the paper is organized as follows: In Section 2, the system-level synthesis problem is introduced. In Section 3, the project scheduling model for the system-level synthesis problem is described. Following the original EDA introduced in Section 4, the HPAEDA is presented in details in Section 5. An example of a video codec design based on the H.261 image compression standard is provided in Section 6. Finally, the paper is ended with some conclusions and future work in Section 7.

2. System-level synthesis problem

The system-level synthesis problem (SLSP) can be described using the "double roof" model, which is illustrated in Fig. 1.

The "double proof" model (Gerstlauer et al., 2009) describes the top-down hardware and software design process of electronic system in an ideal case. One side of the roof corresponds to the software design process; while the other side corresponds to the hardware design process. Both sides contain different abstract layers. A design specification is transformed into an implementation on each abstract layer (vertical arrow). The implementation of each abstract layer is transferred to the next abstract layer as the corresponding design specifications (horizontal arrow).

The SLSP is the top level of the electronic design, which is concerned with how to map the task specification onto the related hardware architecture so as to provide an electronic design specification with high performance and low cost. The specification of electronic design contains three parts (Blickle et al., 1998):

(1) Software behavioral design specification. The software specification of an electronic system is defined by two kinds of tasks, i.e., function tasks and communication tasks. The function task defines the related function module of the electronic system, and the communication task defines the data flow between the function modules.
(2) Hardware architecture design specification. The hardware architecture consists of different hardwares, such as Reduced Instruction Set Chip (RISC), Digital Signal Processor (DSP), Application Specific Integrated Circuit (ASIC), BUS, Random-access Memory (RAM), and so on. Each hardware $k = 1, 2, \ldots, K$ has a cost c_k.
(3) Mapping. The mapping between software and hardware can be defined by a set of Boolean functions. If $map(j,k) = 1$, the task j can be carried out on hardware k; else, the task j cannot be carried out on hardware k. Additionally, the delay function $lt(j,k)$ defines the delay when task j is carried out on hardware k, which is the execution time of task j on hardware k.

The goal of the SLSP is to find an implementation of electronic system to minimize the hardware cost and the system delay (Teich, 2000). The implementation of electronic system can be defined by the hardware implementation and the binding implementation.

(1) Hardware implementation: the hardware set to carry out the electronic system.
(2) Binding implementation: the binding between hardware and software, which is how to choose hardware to carry out the software tasks.

An example of the SLSP is illustrated in Fig. 2 by slightly modifying the example in Blickle et al. (1998). The left hand of Fig. 2 is the design specification of the SLSP. The behavioral design specification is described by a directed graph containing 7 nodes. The white nodes represent function tasks, and the gray nodes represent communication tasks. The hardware architecture contains one RISC, one DSP, one ASIC, and two BUS (BUS1 and BUS2). The arcs between hardware and software define the mapping. For example, task 7 can be executed on any chip (RISC, DSP, ASIC); task 1 can only executed on RISC. The number on each arc is the related delay time. The right hand of Fig. 2 is an electronic system implementation. This implementation adopts all the hardware except BUS2. All the communication tasks are carried out on BUS1.

3. Project scheduling model

The resource-constrained project scheduling problem (RCPSP) is concerned with single-item or small batch production where scarce resources have to be allocated to dependent activities over time (Brucker, Drexl, Möhring, Neumann, & Pesch, 1999). The RCPSP has many extensions, such as multi-mode RCPSP (Wang & Fang, 2011), multi-objective RCPSP (Ballestín & Blanco, 2011), stochastic RCPSP (Ballestín, 2007). The RCPSP comes from practice. The construction of Maya temples in Central and South America and the pyramids of ancient Egypt can be considered as the earliest project scheduling problem (Demeulemeester & Herroelen, 2002). Nowadays, the RCPSP and its extensions widely exist in various industries and service fields, such as medical research experiments

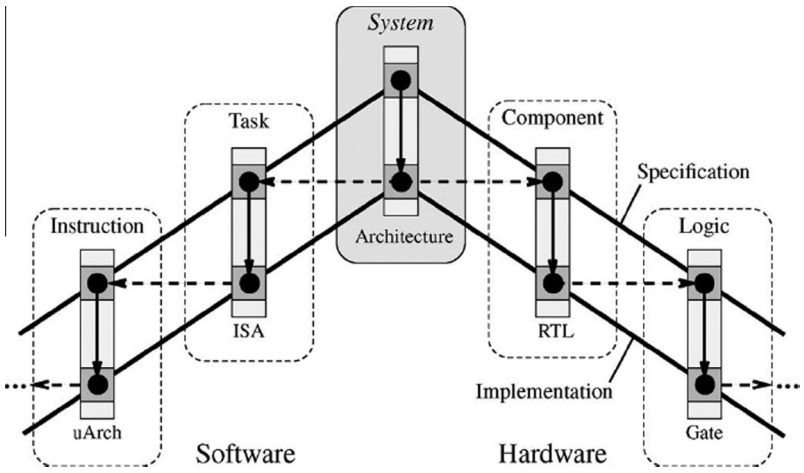

Fig. 1. The "double roof" model (Gerstlauer et al., 2009).

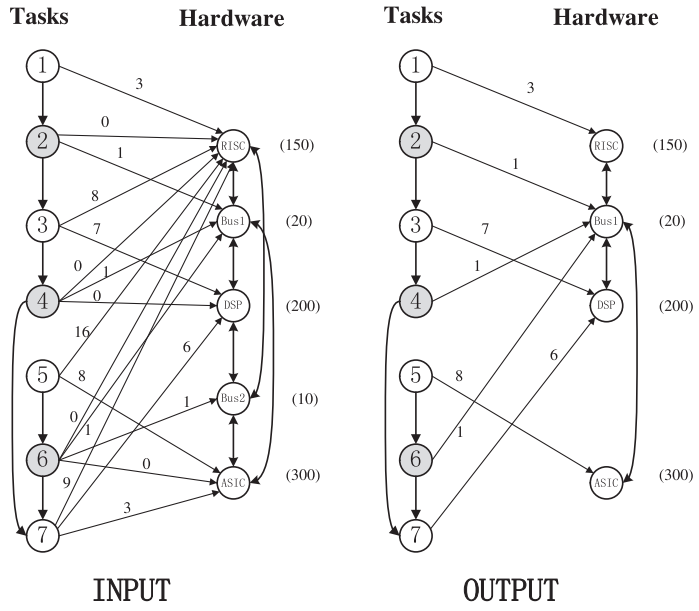

Fig. 2. An example of SLSP (the number in brackets is the cost of corresponding hardware).

scheduling (Hanzálek & Jurcík, 2010), multi-skilled crew scheduling (Liu & Wang, 2012), software project management (Chen & Zhang, 2012), port scheduling (Lorenzoni, Ahonen, & de Alvarenga, 2006), airport scheduling (Kuster, Jannach, & Friedrich, 2009), service centers management (Quintanilla, Pérez, Lino, & Valls, 2012), and grid computing (Mika, Waligora, & Weglarz, 2008).

According to the characteristics of the SLSP, it can be modeled as a **m**ulti-**o**bjective **m**ode-**i**dentity **r**esource-**c**onstrained **p**roject **s**cheduling **p**roblem with **m**ake**s**pan and **r**esource **i**nvestment criteria (MOMIRCPSP-MS-RI). To the best of our knowledge, there is no report work to explore the application of the RCPSP for the system-level synthesis.

3.1. MOMIRCPSP-MS-RI

The MOMIRCPSP-MS-RI is described as follows. A project contains J activities, and each activity $j = 1,\ldots,J$ can be executed in M_j modes. There are precedence constraints between activities, which are defined by the immediate predecessor sets. An immediate predecessor set P_j defines the set of activities, which should be completed before activity j can start. There are K kinds of renewable resources. For each renewable resource $k = 1,2,\ldots,K$, the available amount at each time instant is R_k, and the cost of resource k is c_k. The duration of activity j is d_{jm} if the activity is executed in mode m. In the duration of activity j, r_{jmk} units of renewable resource k are occupied. There are mode-identity constraints, which can be defined by mode-identity sets H_u, where $u = 1,\ldots,U$. The activities belonging to the same mode-identity set should be executed in the same mode. The start and end of the project are represented by the dummy activities 0 and $J + 1$, respectively. The dummy activities do not occupy any resources and their durations are zero. Denote the activity set $\{0,\ldots,J+1\}$ as J^+. The goal of the MOMIRCPSP-MS-RI is to minimize the makespan and resource investment simultaneously.

The mathematical model of the MOMIRCPSP-MS-RI can be described as follows:

$$Minimize\ [MS, RI] \tag{1}$$

$$\text{Subject to} \sum_{m=1}^{M_j} \sum_{t=EFT_j}^{LFT_j} x_{jmt} = 1, \quad j \in J^+ \tag{2}$$

$$\sum_{m=1}^{M_h}\sum_{t=EFT_h}^{LFT_h} t\cdot x_{hmt} \leqslant \sum_{m=1}^{M_j}\sum_{t=EFT_j}^{LFT_j}(t-d_{jm})x_{jmt}, \quad j\in J^+;\ h\in P_j \tag{3}$$

$$r_k(t)\leqslant R_k, \quad k=1,2,\ldots,K;\ t=1,2,\ldots,T \tag{4}$$

$$\sum_{m=1}^{M_h} m\cdot\sum_{t=EF_h}^{LF_h} x_{hmt} = \sum_{m=1}^{M_j} m\cdot\sum_{t=EF_j}^{LF_j} x_{jmt}, \quad u=1,\ldots,U;\ h,j\in H_u \tag{5}$$

$$x_{jmt}=\begin{cases}1, & \text{if activity } j \text{ is performed in mode } m \text{ and finished at time } t\\ 0, & \text{otherwise}\end{cases}, \quad j\in J^+ \tag{6}$$

where Eq. (1) defines that the goal is to minimize the makespan and resource investment; Eq. (2) guarantees that each activity is executed exactly once; Eq. (3) confirms that the precedence constraints are satisfied; Eq. (4) represent the renewable resource constraints; Eq. (5) defines the mode-identity constraints; LFT_j is the latest finish time of activity j; EFT_j is the earliest finish time of activity j; T is an upper bound on the project makespan.

The first objective, makespan MS, can be calculated as follows:

$$MS=\sum_{t=EFT_{J+1}}^{LFT_{J+1}} t\cdot x_{J+1,1,t} \tag{7}$$

The second objective, resource investmentRI, can be calculated as follows:

$$RI=\sum_{k\in K^\rho} c_k \max_{t\in[0,T]}\{r_k(t)\} \tag{8}$$

where $r_k(t)$ is the amount of resource k occupied by the project at time instant t, which can be calculated as follows:

$$r_k(t)=\sum_{j=1}^{J}\sum_{b=\max\{t,EFT_j\}}^{\min\{t+d_j-1,LFT_j\}} r_{jmk}x_{jmb} \tag{9}$$

For multi-objective optimization, the concept of Pareto optimality is often adopted to represent the quality of the solutions. For a minimization problem, the Pareto optimality can be defined as follows: Consider two decision vectors $a,b\in X$, a dominate b ($a\succ b$) iff

$$\forall i\in\{1,2,\ldots,n\},\ f_i(a)\leqslant f_i(b)\wedge\exists j\in\{1,2,\ldots,n\},\ f_j(a)<f_j(b) \tag{10}$$

where X is the decision space; f_i is the ith objective.

The Pareto optimal (non-dominated) solutions are the solutions not dominated by any other feasible decision vectors of the problem. All the Pareto optimal solutions constitute the Pareto set.

An example of the MOMIRCPSP-MS-RI with 5 activities (including two dummy activities $j=0$ and $j=4$) is illustrated in Fig. 3. The project is represented by an activity on node (AoN) network, in which a node represents an activity and an arc represents a precedence constraint between two activities. Each non-dummy activity has two executing modes. The resource requirements and corresponding durations are shown in Table 1. All the non-dummy activities are included in the same mode-identity set (i.e. activities 1, 2, and 3 should be executed in the same mode). One kind of renewable resource with the available amount 6 (i.e. $R=6$) is considered in this example. The cost per unit of the renewable resource is 1. The Pareto front of the example is illustrated in Fig. 4.

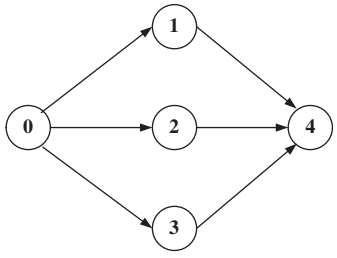

Fig. 3. An example of the MOMIRCPSP-MS-RI.

3.2. Activities and temporal constraints

The behavioral specification of an electronic system can be modeled as a project. Accordingly, each task $j=1,2,\ldots,J$ is treated as an activity. The data interdependencies between tasks are modeled as the precedence constraints. We model the behavioral specification of Fig. 2 as an example. The AoN network of the project is illustrated in Fig. 5. Two dummy activities ($j=0$ and $j=8$) are adopted to represent the start and the end of the project. The dummy activities have zero duration.

3.3. Modes, durations, and resource constraints

In a multi-mode project, each activity has several execution modes. Each activity with a different mode has a related duration and resource requirement. In the system-level synthesis, tasks can be bounded to different hardwares, and different bindings have different latencies. This characteristic motivates us to adopt the multi-mode model. The different hardware can be modeled as different renewable resource $k=1,2,\ldots,K$. The total available amount of each resource R_k is defined as 1, since only one task can be executed on the hardware k at the same time. Accordingly, different bindings can be modeled as different modes. Each activity j can be executed in mode $m\in\{1,2,\ldots,M_j\}$, where $M_j=|\{k|map(j,k)=1, k=1,2,\ldots,K\}|$ is the number of possible mappings of task j. For each mode m, the duration is $d_{jm}=lt(j,k)$, where k is the hardware (renewable resource) to carry out task j in mode m; and r_{jmk} units of renewable resource k is consumed, where $r_{jmk}\in\{0,1\}$. Additionally, the dummy activities do not request any resources. The modes, durations, and renewable resource constraints of the system-level synthesis problem of Fig. 2 are shown in Table 2.

It is worth noting that there are mode-identity constraints between some communication tasks and function tasks. For example, the communication task 2 can be executed on RISC only if function task 1 and 3 are both executed on RISC.

3.4. Schedule generating scheme

For the project scheduling problem, the schedule generating scheme (SGS) is often adopted to construct a non-iterative schedule. The SGS assigns each activity j of the problem graph a start time $ST(j)$ such that the overall makespan is minimized. Additionally, all precedence constraints and resource constraints must be satisfied. In this paper, the parallel SGS (PSGS) (Kolisch, 1996) is adopted to generate the non-iterative schedule. The generated schedule corresponding to the output of system-level synthesis problem of Fig. 2 is shown in Fig. 6. The mode of each activity $j=1,2,\ldots,J$ is 1, 2, 2, 2, 2, 2, and 2, respectively. The resource

Table 1
Renewable resource requirements and corresponding durations.

Activity	Mode 1		Mode 2	
	d_{jm}	r_{jmk}	d_{jm}	r_{jmk}
1	3	1	2	2
2	3	1	2	2
3	2	1	1	2

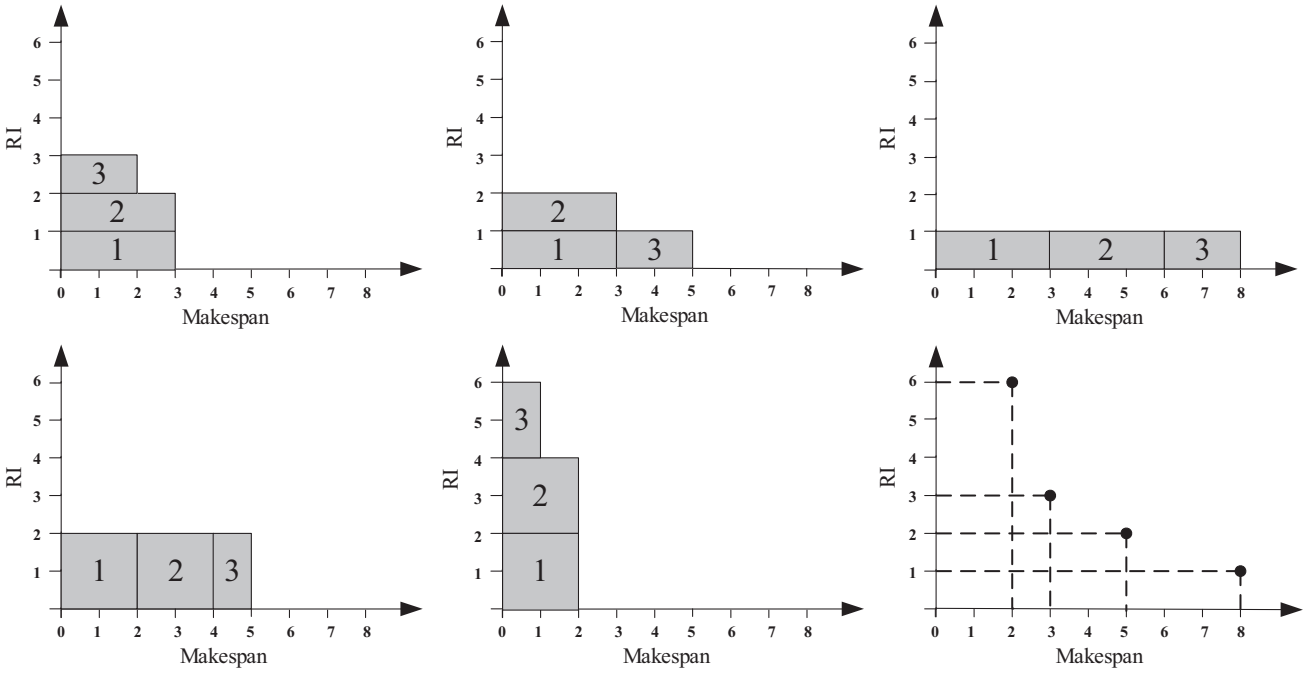

Fig. 4. Pareto solutions.

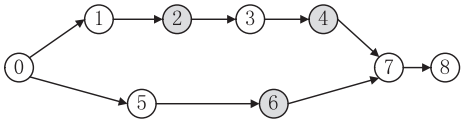

Fig. 5. The AoN network of the behavioral specification of Fig. 2.

Table 2
The modes, durations, and resource requirements of the SLSP in Fig. 2.

Activity	#modes	mode	duration	Renewable resource requirements				
				RISC	Bus1	DSP	Bus2	ASIC
0	1	1	0	0	0	0	0	0
1	1	1	3	1	0	0	0	0
2	2	1	0	1	0	0	0	0
		2	1	0	1	0	0	0
3	2	1	8	1	0	0	0	0
		2	7	0	0	1	0	0
4	3	1	0	1	0	0	0	0
		2	1	0	1	0	0	0
		3	0	0	0	1	0	0
5	2	1	16	1	0	0	0	0
		2	8	0	0	0	0	1
6	4	1	0	1	0	0	0	0
		2	1	0	1	0	0	0
		3	1	0	0	0	1	0
		4	0	0	0	0	0	1
7	3	1	9	1	0	0	0	0
		2	6	0	0	1	0	0
		3	3	0	0	0	0	1
8	1	1	0	0	0	0	0	0

investment $RI = 150 + 20 + 200 + 300 = 670$ and the makespan $MS = 18$.

The PSGS can schedule a non-cyclic task graph to generate a non-iterative schedule. However, the iterative (periodic) schedule is usually of interest in a data flow-dominant system due to the fact that the tasks specified by the problem graph are executed repeatedly. The iterative schedule is characterized by the fact that the schedule repeats after a certain time (called period P). The iterative scheduling algorithm developed by Blickle (1996) is often used to generate the iterative schedules. Its basic idea is to remove edges (dependencies) from the problem graph, such that the non-iterative scheduler can be used. In our procedure, the ISA-PSGS is proposed by combining the iterative scheduling algorithm and PSGS to generate the iterative schedules. The ISA-PSGS algorithm works as follows:

First, a non-iterative schedule is obtained using the PSGS. In this initial try, all operations are in the same iteration interval $\varphi(j) = 1$, $j = 1, \ldots, J$. Then all activities that end later than the minimum period $P_{\min}$ are moved into next iteration interval. "Moving" an activity j corresponds to the deletion of all precedence constraints $e \in \{(i,j) | FT(i) \leqslant P_{\min} \wedge FT(j) > P_{\min}\}$. As the mode of each activity has already been fixed, $P_{\min}$ can be calculated as follows:

$$P_{\min} = \max_{k \in \{1,2,\ldots,K\}} \sum_{j \in \{1,2,\ldots,J\} \wedge r_{jk}=1} d_{jm} \tag{11}$$

The new problem graph (with some edges removed) is scheduled again using the PSGS. The current period P of the iterative schedule is then equivalent to the makespan of the schedule according to the new problem graph. A parameter $\varphi_{\max}$ is used to limit the maximum number of iteration intervals. This procedure repeats until the current period is not smaller than the previous one or equal to $P_{\min}$. The pseudo code of the procedure is given in Fig. 7.

For the example of Fig. 2, we show the procedure of the ISA-PSGS algorithm. The minimum period is calculated as follows:

$$P_{\min} = \max\{3, 1+1+1, 7+6, 8\} = 13 \tag{12}$$

Consider the non-iterative schedule shown in Fig. 6, the activity 7 is moved into next iteration according to the ISA-PSGS. That is, the edges (4,7) and (6,7) are removed from the problem graph of Fig. 5. If the modified problem graph is scheduled again, the schedule of Fig. 8 is obtained. In the next step, activity 4 is moved into next iteration, i.e. the edge (3,4) is removed. The schedule of Fig. 9 is obtained. The corresponding iterative schedule is shown in

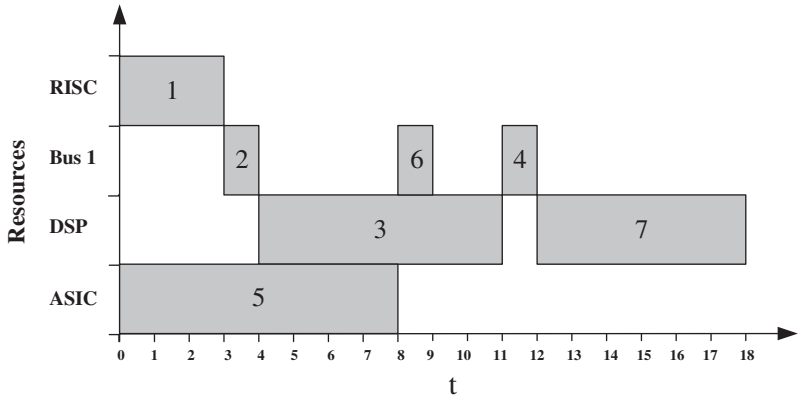

Fig. 6. The non-iterative schedule corresponding to the output of Fig. 2.

Algorithm: ISA-PSGS

Input: activity-mode-priority-resource list *AMPRL* (see Section 5.1), Edge set of AoN network E, maximum number of iterations $\varphi_{\max}$

Output: the iterative schedule τ

Begin

$\tau' = PSGS(AMPRL, E)$;

$P_{\min} = \max_{k\in\{1,2,\ldots,K\}} \sum_{j\in\{1,2,\ldots,J\}\wedge r_{jk}=1} d_{jm}$

DO

$\tau = \tau'$;

For each precedence constraint (i, j)

if $FT(i) \le P_{\min}$ and $FT(j) > P_{\min}$ and $\varphi(j)<\varphi_{\max}$

$\varphi'(j) = \varphi(j)+1$;

Remove the edge $E = E\setminus\{(i, j)\}$;

$\tau' = PSGS(AMPRL, E)$;

While $P(\tau') < P(\tau)$ and $P(\tau') > P_{\mu v}$

If $P(\tau') = P_{\min}$

return τ';

Else

Return τ;

END

Fig. 7. The pseudo code of ISA-PSGS.

Fig. 10. As the period of this schedule is optimal, the calculation is finished.

4. Brief introduction to EDA

Estimation of distribution algorithm (EDA) (Larrañaga & Lozano, 2002; Lozano, Larrañaga, Inza, & Bengoetxea, 2006) is a general framework of statistical learning based optimization algorithm. With the help of probability model, the EDA makes the population movement track the promising search area. For a minimization problem, the procedure of the EDA can be stated as follows.

Step 1: Initialization phase. Initialize the probability model, and set $g = 0$.

Step 2: Sampling phase. Generate a new population $X^g = [x_1^g, x_2^g, \ldots, x_N^g]$ with N individuals by sampling the probability model. Each individual $x_i^g = [x_{i,1}^g, x_{i,2}^g, \ldots, x_{i,D}^g]$ represents a solution.

Step 3: Elite selection phase. Calculate the fitness value of each individual x_i^g $(i = 1,2,\ldots,N)$, and sort individuals in an ascending order of the fitness values. Then, select Q best individuals to form the elite set $X_{Elite}^g = [x_1^g, x_2^g, \ldots, x_Q^g]$.

Step 4: Updating phase. Update the probability model with the elite set X_{Elite}^g as follows:

$$prob(x^{g+1}) = prob(x^g | X_{Elite}^g) \quad (13)$$

Step 5: Stopping condition checking phase. If the stopping condition is met, output the best individual and its fitness value; else, set $g = g + 1$, and go to step 2.

The core of the EDA procedure is to estimate the probability distribution. Due to the different types of problem, different probability models should be designed to estimate the underlying probability distribution. In the next section, a special probability model and an updating mechanism for the EDA will be presented to solve the particular SLSP.

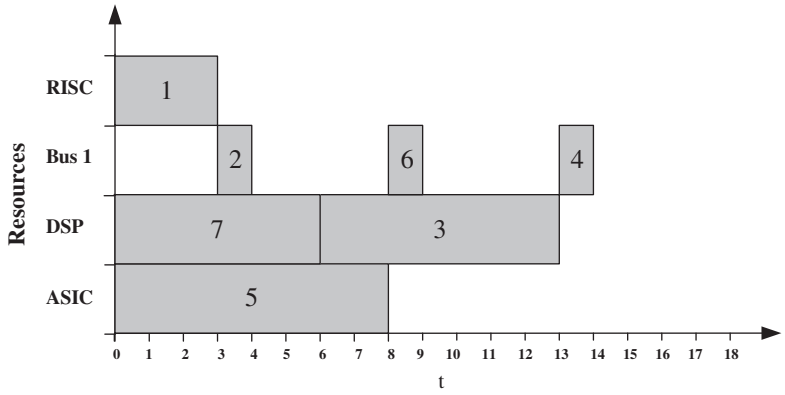

Fig. 8. The schedule after removing edges (4,7) and (6,7).

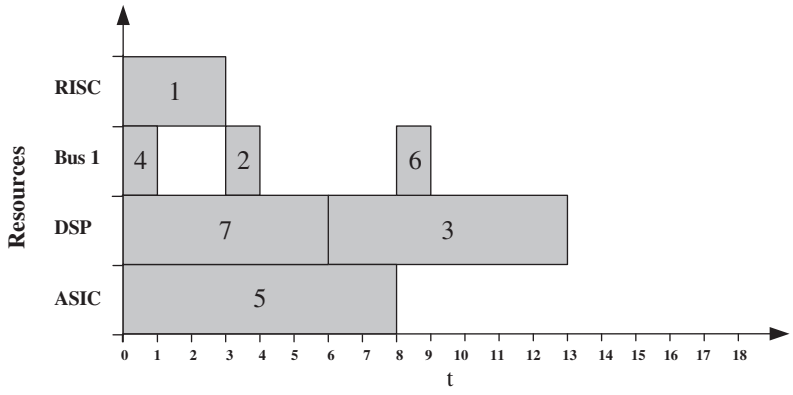

Fig. 9. The schedule after removing edges (4,7), (6,7), and (3,4).

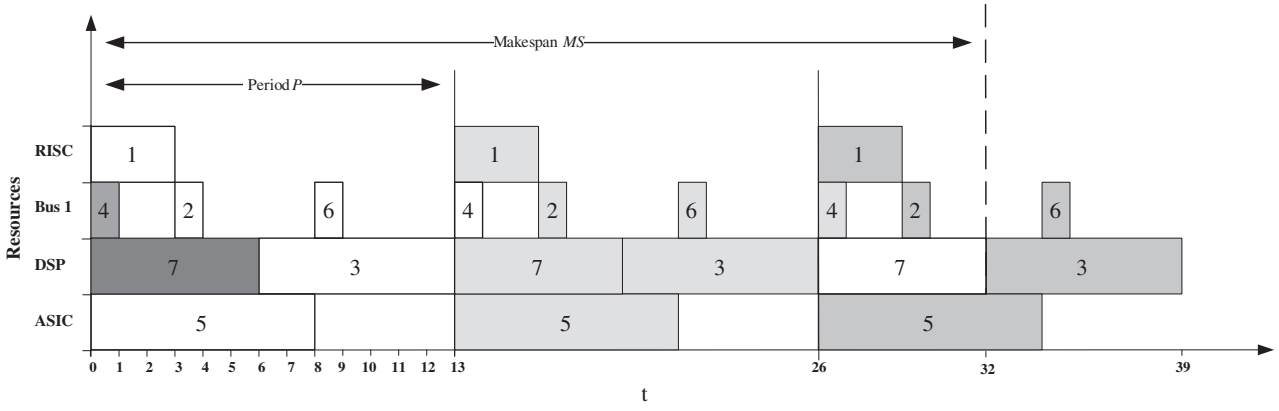

Fig. 10. The iterative schedule for the output of Fig. 2.

5. The proposed HPAEDA

In this section, the encoding scheme, hybrid probability model, sampling mechanism, and updating mechanism will be introduced one by one. Then, the framework of the HPAEDA will be presented.

5.1. Encoding scheme

In the HPAEDA, the activity-mode-priority-resource list (AMPRL) based encoding scheme is adopted, which contains four parts: (1) an activity list (AL) $\pi = (\pi_1, \pi_2, \ldots, \pi_J)$; (2) a model assignment list (ML) $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_K)$, where $\lambda_j \in \{1, \ldots, M_j\}$; (3) a resource priority list (PL) $\beta = (\beta_1, \beta_2, \ldots, \beta_J)$; (4) a resource list (RL) $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_K)$, where $\gamma_k \in \{0, 1\}$. Each element π_j in the AL represents the activity with a priority order j. Each element λ_j in the ML indicates the mode of the activity j. If the element $\gamma_k = 1$, the resource k is available; otherwise, the resource will not be used. The PL defines the priority order of the resource. If an additional resource is needed, the unused resource with the highest priority order will be added. Thus, $AMPRL = [\pi, \lambda, \beta, \gamma]$.

5.2. Hybrid probability model

It is a key issue to construct the probability model for designing the EDA (Lozano et al., 2006). Here, a hybrid probability model is used to predict the promising search area of the MOMIRCPSP-

MS-RI. The hybrid probability model is composed of three probability matrixes: *Prob_res*(*g*), *Prob_act*(*g*), and *Prob_mod*(*g*).

(1) *Prob_act*(*g*): it is used to predict the position of each activity in the *AL*.

$$Prob\_act(t) = \begin{pmatrix} \alpha_{11}(g) & \cdots & \alpha_{1J}(g) \\ \vdots & \ddots & \vdots \\ \alpha_{J1}(g) & \cdots & \alpha_{JJ}(g) \end{pmatrix} \tag{14}$$

where the element $\alpha_{ji}(g)$ represents the probability that the activity j is placed at position i of the *AL* at generation g. That is, $\alpha_{ji}(g)$ is an index to indicate the satisfactory degree to place activity j at position i. This matrix is initialized as Eq. (15) to ensure that the whole solution space can be sampled uniformly.

$$Prob\_act(0) = \begin{pmatrix} 1/J & \cdots & 1/J \\ \vdots & \ddots & \vdots \\ 1/J & \cdots & 1/J \end{pmatrix} \tag{15}$$

(2) *Prob_res*(*g*): A similar matrix is used to generate the PL, and it is initialized as Eq. (17).

$$Prob\_res(g) = \begin{pmatrix} \eta_{11}(g) & \cdots & \eta_{1K}(g) \\ \vdots & \ddots & \vdots \\ \eta_{K1}(g) & \cdots & \eta_{KK}(g) \end{pmatrix} \tag{16}$$

$$Prob\_res(0) = \begin{pmatrix} 1/K & \cdots & 1/K \\ \vdots & \ddots & \vdots \\ 1/K & \cdots & 1/K \end{pmatrix} \tag{17}$$

(3) *Prob_res*(*g*): It is used to predict the mode adopted by each activity.

$$Prob\_mod(g) = \begin{pmatrix} \mu_{11}(g) & \cdots & \mu_{1M}(g) \\ \vdots & \ddots & \vdots \\ \mu_{J1}(g) & \cdots & \mu_{JM}(g) \end{pmatrix} \tag{18}$$

where $M = \max_{j=1,2,\ldots,J} M_j$, the element μ_{ji} represents the probability that the activity j adopt mode i at generation g. That is, $\mu_{ji}(g)$ is an index to indicate the satisfactory degree for activity j to be carried out in mode i. This matrix is initialized as Eq. (19) to ensure that the whole solution space can be sampled uniformly.

$$\mu_{ji}(0) = \begin{cases} 1/M_j & \text{if } i \leqslant M_j \\ 0 & \text{else} \end{cases} \tag{19}$$

Note that μ_{ji} is set as zero if the mode i is not available for activity j.

5.3. Sampling mechanism

At every position i, the selection probability of activity j, i.e. Pa_{ji}, is calculated according to probability matrix *Prob_act*(*g*) over the set of eligible activities *SE* as follows:

$$Pa_{ji} = \alpha_{ji} \Big/ \sum_{h \in SE} \alpha_{hi} \tag{20}$$

If activity j has already been placed at some positions, the whole line $\alpha_{j1}, \alpha_{j2}, \ldots, \alpha_{jJ}$ of probability matrix *Prob_act* is set zero. As a result, each activity will be selected only once in the *AL*. According to selection probability of each activity, the *AL* will be generated by selecting activity one by one. The PL can be generated in a similar way. For the order i, the selection probability of a resource j, i.e. Pr_{ji}, is calculated according to probability matrix *Prob_res*(*g*) over the set of resources as follows:

$$Pr_{ji} = \eta_{ji} \Big/ \sum_{h \in \{1,2,\ldots,K\}} \eta_{hi} \tag{21}$$

The *RL* is simple and easy to use. However, how to generate a feasible RL is a difficult problem. The coding might result in infeasible solutions easily, since not all the needed resources are available sometimes. In order to repair an infeasible solution, requisite resources need to be added. The order of the additional resource has a large influence on the quality of the solution. Here, the resource priority list is adopted to define the order of resources. The procedure to generate a feasible RL is shown in Fig. 11.

Procedure **Resource List Generation** (β)
- Generate a random integer $a \in [0, K]$
- $\gamma = \{0\}$;
- For ($i = 0,\ldots,a$)
  - $\gamma_{\beta_i} = 1$;
- For ($i = 0,\ldots,J$)
  - If the activity i is not executable in any modes with γ
    - For ($j = a+1,\ldots,K$)
      - If $\gamma_{\beta_j} == 0$
        - $\gamma' = \gamma$;
        - $\gamma'_{\beta_j} = 1$;
      - If the activity i becomes executable in at least one mode with γ'
        - $\gamma = \gamma'$;
        - break;
- Return γ;

Fig. 11. The RL generating procedure.

Then, a mode for each activity in the AL is chosen with the following mode selection probability. After generating the AMRPL, the ISA-PSGS is used to transform it to a schedule.

$$Pm_{ji} = \mu_{ji} \Big/ \sum_{h \in F_j} \mu_{jh} \tag{22}$$

where F_j is the feasible mode set of activity j with the available resources defined by the RL.

5.4. Pareto archive and updating archive

In the HPAEDA, a Pareto archive *PA* and an updating archive *UA* are used to store the non-dominated solutions that have been generated and the newly found non-dominated solutions, respectively. At each generation, each individual I_j ($j = 1,2,\ldots,|POP|$) is compared with each member of the Pareto archive A_j ($j = 1,2,\ldots,|PA|$). Then, the newly generated solutions that dominate the Pareto archive are added into *PA*. All these newly found non-dominated solutions are preserved in *UA*. Meanwhile, the solutions of *PA* dominated by the newly found non-dominated solutions will be discarded. The procedure of archive updating is illustrated in Fig. 12.

5.5. Updating mechanism

In this paper, the probabilistic matrixes *Prob_act*(*g*), *Prob_res*(*g*) and *Prob_mod*(*g*) are updated by the population-based updating mechanism. After updating the archives, the newly generated non-dominated solutions in *UA* are adopted to update the hybrid probability model as follows:

$$Prob\_act_{ji}(g+1) = (1-\theta)\cdot Prob\_act_{ji}(g) + \frac{\theta}{|UA|}\sum_{k=1}^{|UA|} I_{ji}^{k}, \quad (1 \leqslant i,\ j \leqslant J) \tag{23}$$

$$Prob\_res_{ji}(g+1) = (1-\theta)\cdot Prob\_res_{ji}(g) + \frac{\theta}{|UA|}\sum_{k=1}^{|UA|} Q_{ji}^{k}, \quad (1 \leqslant i,\ j \leqslant K) \tag{24}$$

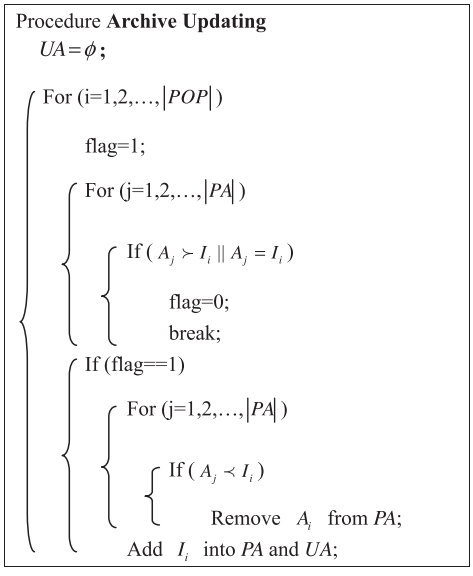

Fig. 12. Procedure of archive updating.

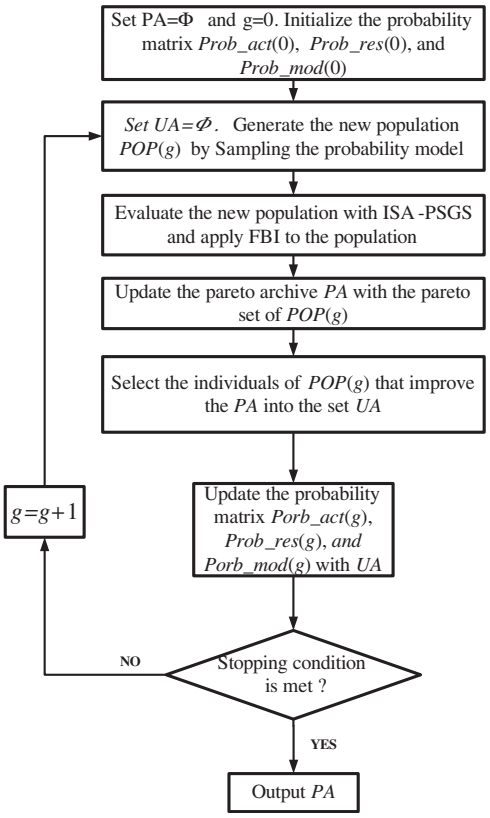

Fig. 13. The framework of HPAEDA.

$$Prob\_mod_{ji}(g+1) = (1-\theta)\cdot Prob\_mod_{ji}(g) + \frac{\theta}{|UA|}\sum_{k=1}^{|UA|} R_{ji}^{k}, \quad (1 \leqslant i \leqslant M,\ 1 \leqslant j \leqslant J) \tag{25}$$

where θ is the learning speed, and I_{ji}^{k}, Q_{ji}^{k} and R_{ji}^{k} are the indicator functions. Note that the hybrid probability model will not be updated if $UA = \phi$.

The indicator functions are defined as follows:

$$I_{ji}^{k} = \begin{cases} 1 & \text{if activity } j \text{ is placed at position } i \text{ in the } k\text{th individual of } UA \\ 0 & else \end{cases} \tag{26}$$

$$Q_{ji}^{k} = \begin{cases} 1 & \text{if resource } j \text{ has an order } i \text{ in the } k\text{th individual of } UA \\ 0 & else \end{cases} \tag{27}$$

$$R_{ji}^{k} = \begin{cases} 1 & \text{if activity } j \text{ chooses mode } i \text{ in the } k\text{th individual of } UA \\ 0 & else \end{cases} \tag{28}$$

5.6. Procedure of HPAEDA

According to the above design, the HPAEDA for the MOM-IRCPSP-MS-RI is summarized as follows. First, the probability matrixes *Prob_act*(0), *Prob_res*(0), *and Prob_mod*(0) are initialized. Second, the new population *POP*(g) is generated by sampling the hybrid probability model with the special sampling mechanism. After all the individuals are evaluated by ISA-PSSGS, the well-known forward–backward improvement (FBI) (Li & Willis, 1992) is adopted to improve each individual in *POP*(g). Then, the Pareto

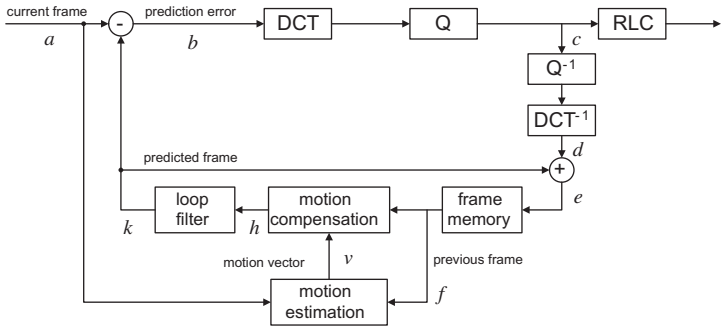

Fig. 14. The block diagram of H.261 video compression standard.

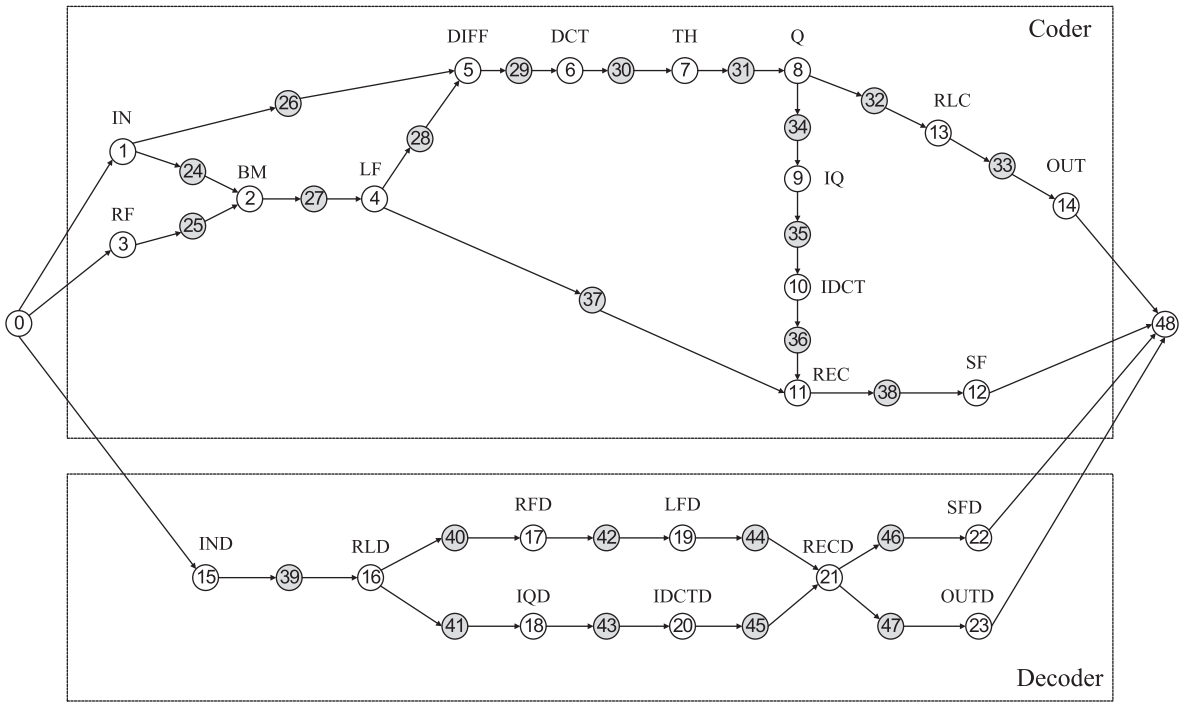

Fig. 15. The AoN network of the behavioral specification in Fig. 6.

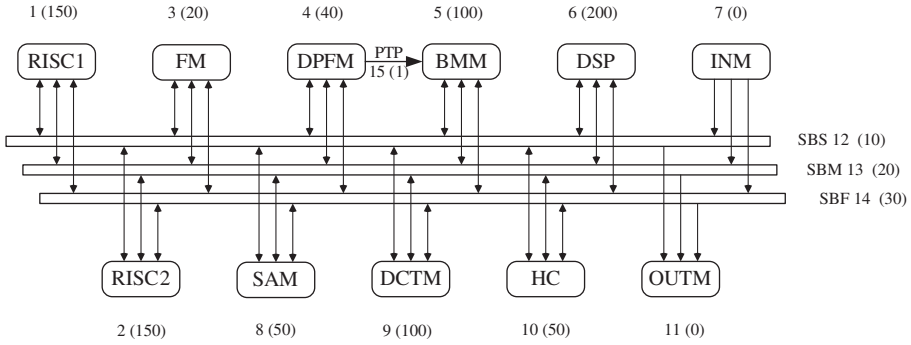

Fig. 16. The hardware architecture to carry out the video codec H.261.

archive *PA* is updated with *POP*(g). After the individuals improving *PA* are selected into *UA*, the probability matrix *Prob_act*(g), *Prob_res*(g), *and Prob_mod*(g) are updated with *UA*. The procedure is repeated until a stopping condition is met.

Straightforwardly, the framework of the proposed HPAEDA is illustrated in Fig. 13.

6. Case study

6.1. Testing problem

In this section, the proposed modeling methodology and the HPAEDA will be tested by using the case of a video codec design

based on video compression standard H.261, whose block diagram is shown in Fig. 14 (Bovik, 2005).

The basic idea is to adopt both intraframe and interframe coding mode for improving the efficiency of the codec. H.261 operations on macroblocks with four 8×8 pixels of brightness information and two 8×8 pixels of chromatism information. Each prediction error frame b should be transformed by discrete cosine transform (DCT) and quantizer (Q). The data is further compressed through run-length coder (RLC) before being transformed. After the transformation with inverse quantizer (Q^{-1}) and inverse discrete cosine transform (DCT^{-1}), the uncompressed frame c is added to the predicted frame k and stored in the frame memory. The interframe coding is based on motion estimation, which compares each macroblock of the current frame with the corresponding macroblock of the previous frame. A motion vector v is generated after motion estimation. Only v is transferred to the receiver. The frame h can be reconstructed by motion compensation with motion vector v. In addition, a loop filter can be used to improve the video quality by reducing the high-frequency noise.

The behavioral specification can be modeled as a project network, as shown in Fig. 15. The project network contains two parts: coder and decoder. Motion estimation and motion compensation is done by the block matching operation (BM), the block subtraction is named DIFF (difference) and block addition is named REC (recover). Additionally, the quantization is split up into threshold calculation (TH) and quantization (Q). The operations are performed on macro blocks, and each communication node represents a transmission of one macro block. However, the block matching operation needs average three macro blocks of the previous frame for its operation. The coder is usually combined with a decoder to obtain a codec implementation. All decoding operations end with the suffix "D". Basically, the decoding consists of the lower part of the loop of the coding algorithm (Fig. 12.). However, the motion compensation vector has to be extracted from the input data after the run-length decoding operation (RLD).

For easy comparison, the hardware architecture used by Blickle et al. (1998) as Fig. 16 is adopted to carry out the function tasks and communication tasks. The hardware architecture contains three

Table 3
The modes, durations, and resource requirements of H.261.

Node #	Oper.	Mode1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
		Res./Dur.	Res./Dur..	Res./Dur..	Res./Dur..	Res./Dur.	Res./Dur.	Res./Dur.
0	Dummy	None/0						
1	IN	INM/0						
2	BM	BMM/22	DSP/60	RISC1/88	RISC2/88			
3	RF	FM/0	DPFM/0					
4	LF	HC/2	DSP/3	RISC1/9	RISC2/9			
5	DIFF	SAM/1	DSP/2	RISC1/2	RISC2/2			
6	DCT	DCTM/2	DSP/4	RISC1/8	RISC2/8			
7	TH	HC/2	DSP/8	RISC1/8	RISC2/8			
8	Q	HC/1	DSP/2	RISC1/2	RISC2/2			
9	IQ	HC/1	DSP/2	RISC1/2	RISC2/2			
10	IDCT	DCTM/2	DSP/4	RISC1/8	RISC2/8			
11	REC	SAM/1	DSP/2	RISC1/2	RISC2/2			
12	SF	FM/0	DPFM/0					
13	RLC	HC/2	DSP/8	RISC1/8	RISC2/8			
14	OUT	OUTM/0						
15	IND	INM/0						
16	RLD	HC/2	DSP/8	RISC1/8	RISC2/8			
17	RFD	FM/0	DPFM/0					
18	IQD	HC/1	DSP/2	RISC1/2	RISC2/2			
19	LFD	HC/2	DSP/3	RISC1/9	RISC2/9			
20	IDCTM	DCTM/2	DSP/4	RISC1/8	RISC2/8			
21	RECD	SAM/1	DSP/2	RISC1/2	RISC2/2			
22	SFD	FM/0	DPFM/0					
23	OUTD	OUTM/0						
24	Comm.	SBF/1	SBM/2	SBS/3				
25	Comm.	SBF/3	SBM/6	SBS/9	PTP/1			
26	Comm.	SBF/1	SBM/2	SBS/3				
27	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	
28	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	
29	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	
30	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	
31	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	HC/0
32	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	HC/0
33	Comm.	SBF/1	SBM/2	SBS/3				
34	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	HC/0
35	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	
36	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	
37	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	
38	Comm.	SBF/1	SBM/2	SBS/3				
39	Comm.	SBF/1	SBM/2	SBS/3				
40	Comm.	None/0						
41	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	HC/0
42	Comm.	SBF/1	SBM/2	SBS/3				
43	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	
44	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	
45	Comm.	SBF/1	SBM/2	SBS/3	DSP/0	RISC1/0	RISC2/0	
46	Comm.	SBF/1	SBM/2	SBS/3				
47	Comm.	SBF/1	SBM/2	SBS/3				
48	Dummy	None/0						

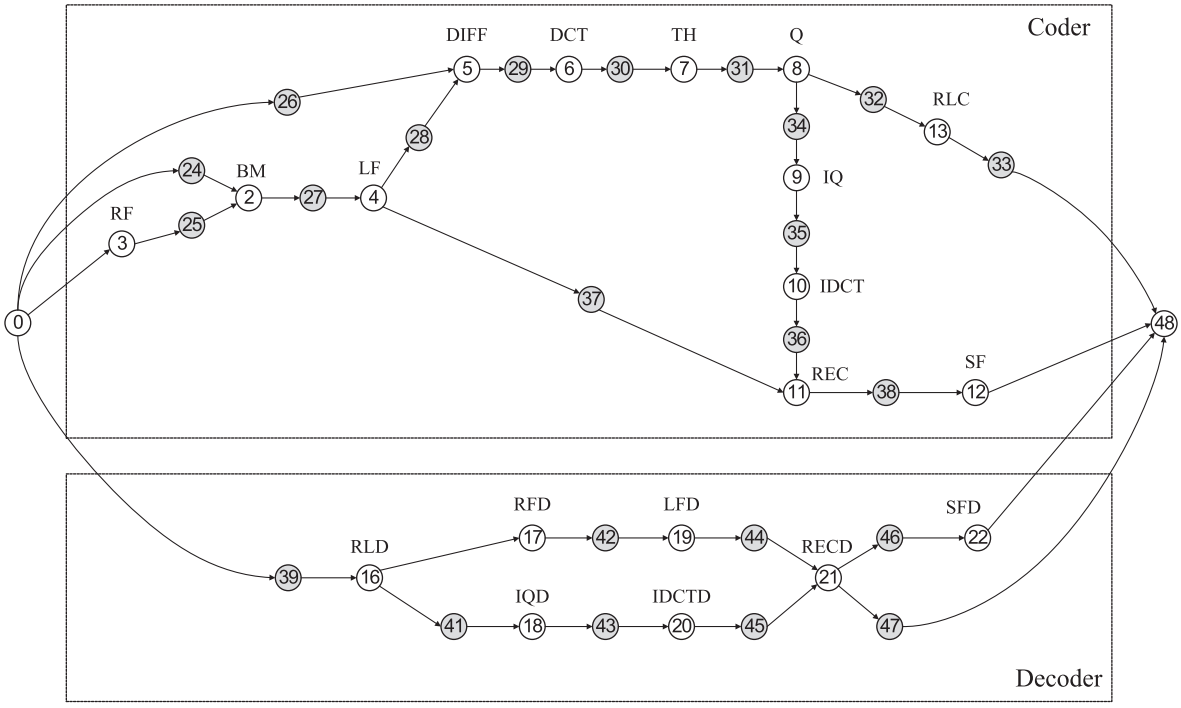

Fig. 17. The simplified AoN network.

Table 4
The Pareto set of different algorithms (Cost, Period).

HPAEDA	SPEA	Single-objective EA	RTS + Pareto ranking
(180,166)	(180,166)	(180,166)	(180,166)
(230,114)	(230,114)	(230,114)	(230,114)
(280,78)	(280,78)	(280,78)	(280,78)
(330,48)	(330,48)	(330,54)	(330,54)
(340,36)	(340,36)	(340,42)	(350,23)
(350,22)	(350,22)	(350,22)	(370,22)

buses with different speed (SBS, SBM, SBF), a storage module (a single-port memory FM and a dual-port memory DPFM), RISC processor (RISC1 and RISC2), some custom hardware modules (one block matching module (BMM), the DCT/IDCT module (DCTM), addition and subtraction module (SAM), Huffman coding module (HC), and input/output module (INM)). Besides, there is a unidirectional point-to-point high-speed bus (PTP) between BMM and DPFM, which can accelerate the speed of reading data from DPFM. The RISC and DSP can execute any tasks of H.261. DSP is faster but more expensive, while RISC is slower but cheaper. Other hardware modules can only deal with specific tasks. For example, the DCTM modules can only deal with DCT, IDCT, and IDCTD. The number in the brackets on each hardware module in Fig. 16 is the corresponding hardware cost.

Based on the modeling methodology described in Section 3, the design problem of H.261 video codec can be modeled as a MOMIRCPSP-MS-RI. The corresponding project information, such as executing mode, renewable resource requirements, and duration, are illustrated in Table 3.

Note that the problem has the mode-identity constraints. The executing mode of some communication activities depends on the two connecting function activities. Those with mode-identity constraints are activities 27–32, 34–36, 41, 43–45. For example, the activity 27 must be executed in mode 4 if activity 2 and 4 are both executed in mode 2.

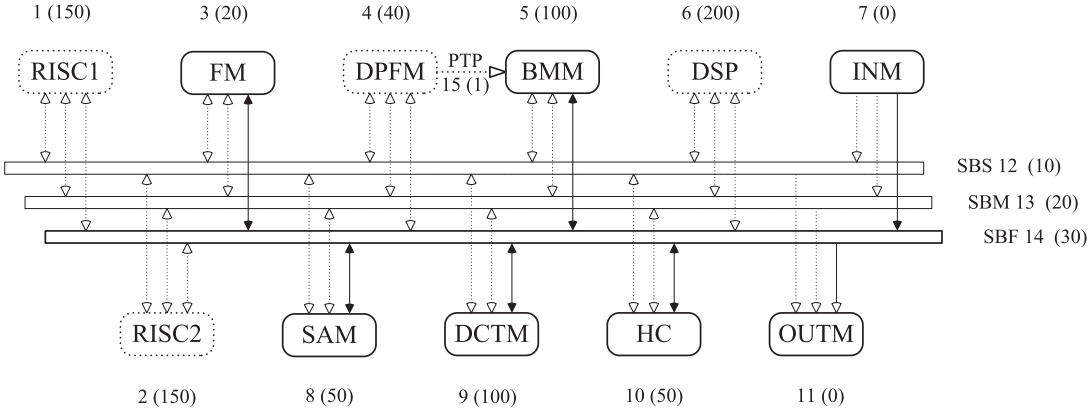

Fig. 18. The hardware implementation for solution (350,22).

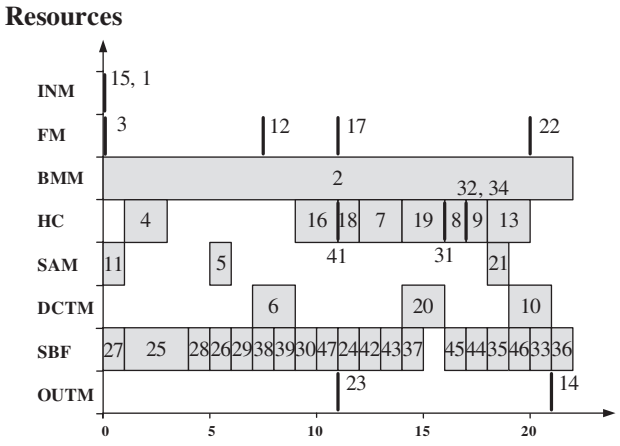

Fig. 19. The Gantt chart for solution (350,22).

6.2. AoN network simplification

Compared to other operations, the executing times of input/output operations (IN, OUT, IND, and OUTD) are smaller. So, it assumes that the corresponding durations are zero. Since the data of motion compensation vector is smaller than macroblock, it assumes that the duration of communication activity 40 is zero. Accordingly, we can delete activities 1, 14, 15, 23, 40 from the AoN network to simplify the project. The simplified AoN network is illustrated in Fig. 17.

6.3. Computational results

The procedure is coded in Visual C++ 2005 and tested on IBM ThinkPad X60 (Core T5600, 1.83 GHz). The HPAEDA is compared with the single-objective EA (Blickle, 1996), SPEA (Zitzler & Thiele, 1999), and RTS + Pareto ranking (Harik, 1995). The experiment setting (Blickle, 1996) is adopted, i.e., running the HPAEDA with a

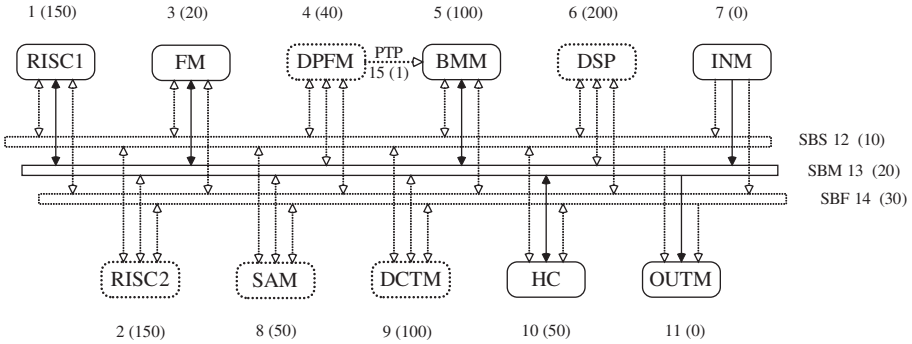

Fig. 20. The hardware implementation for solution (340,36).

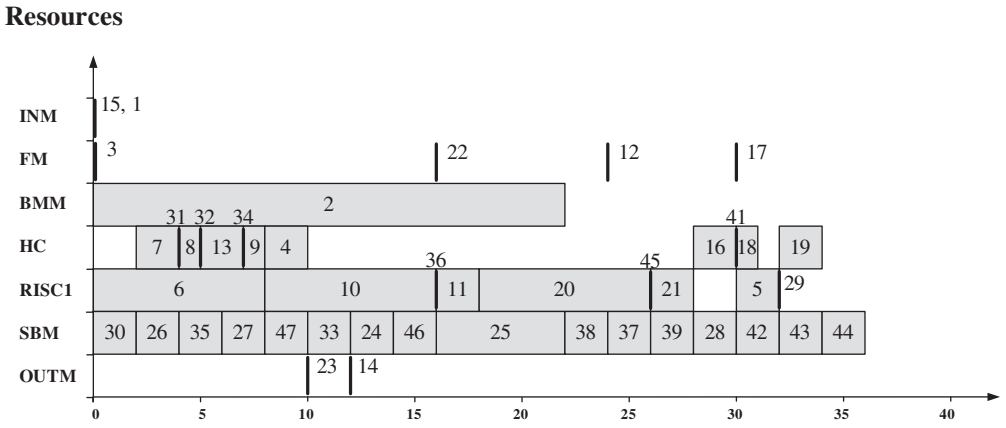

Fig. 21. The Gantt chart for solution (340,36).

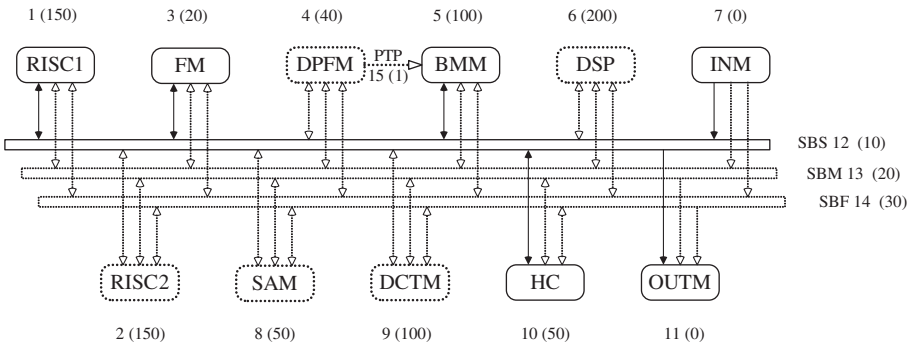

Fig. 22. The hardware implementation for solution (330,48).

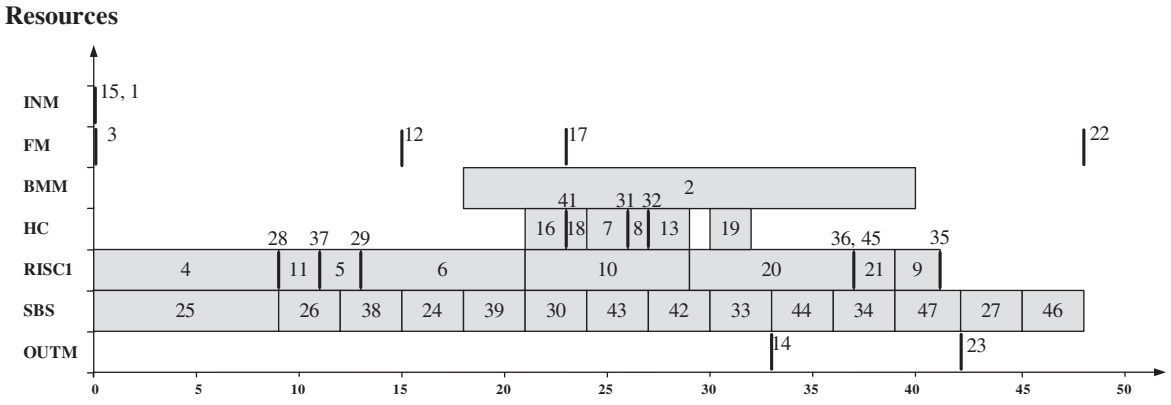

Fig. 23. The Gantt chart for solution (330,48).

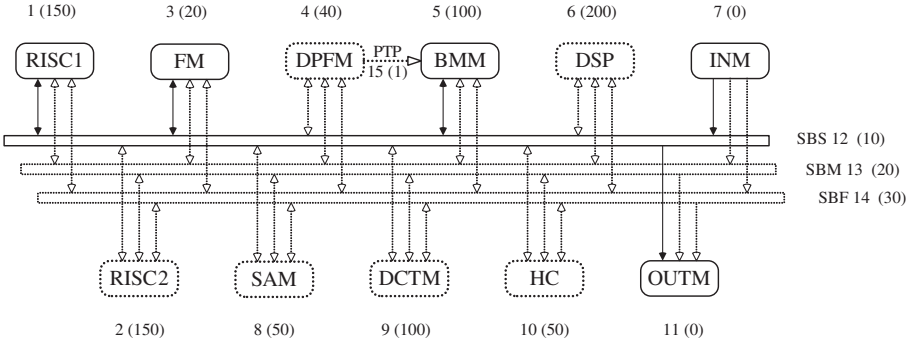

Fig. 24. The hardware implementation for solution (280,78).

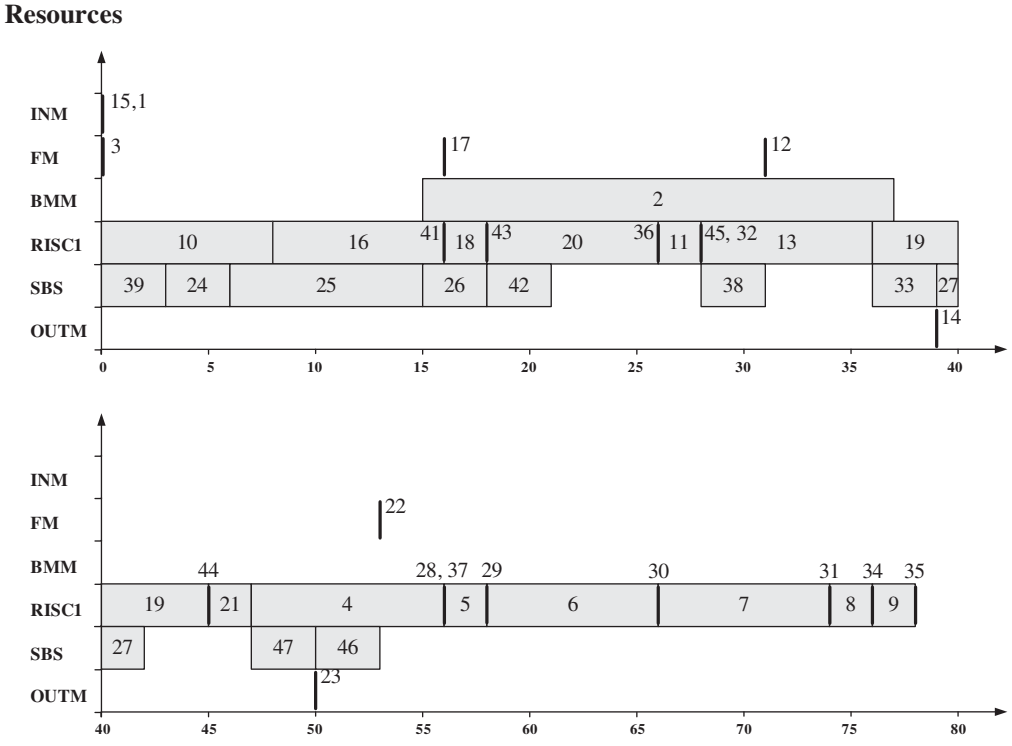

Fig. 25. The Gantt chart for solution (280,78).

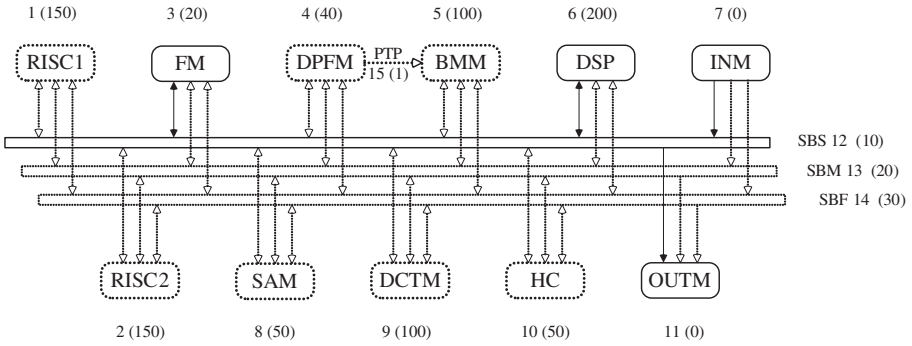

Fig. 26. The hardware implementation for solution (230,114).

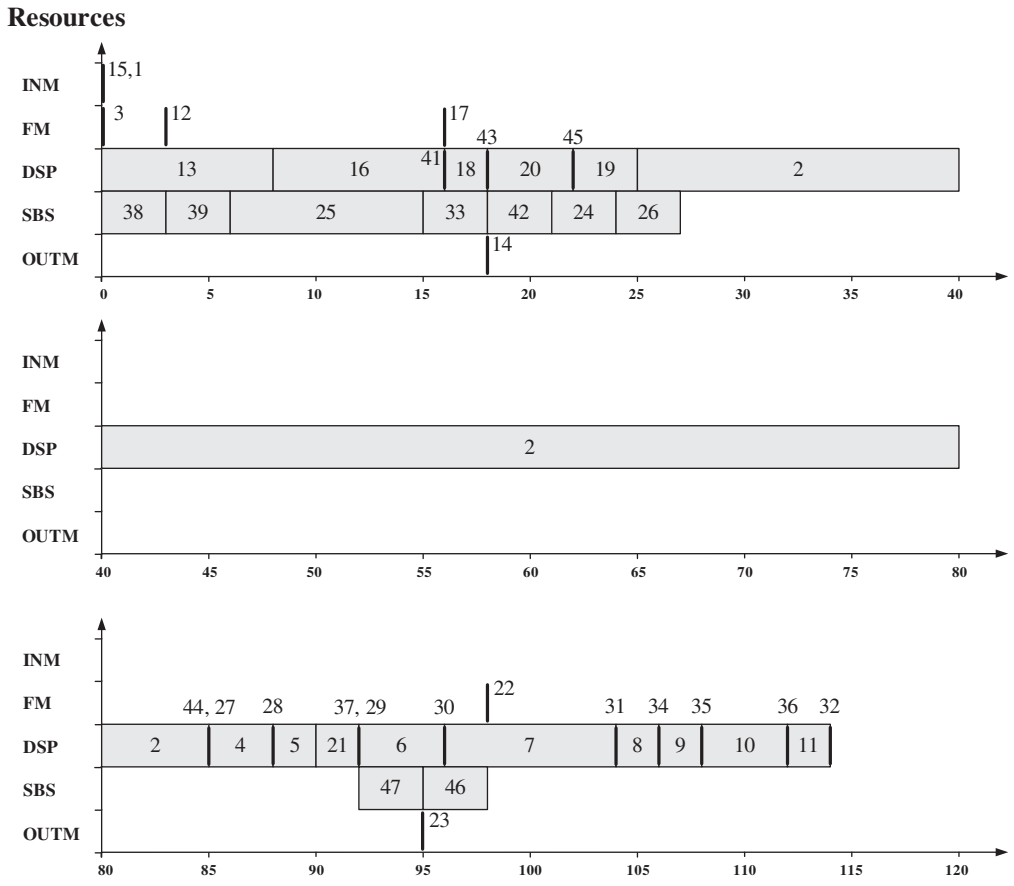

Fig. 27. The Gantt chart for solution (230,114).

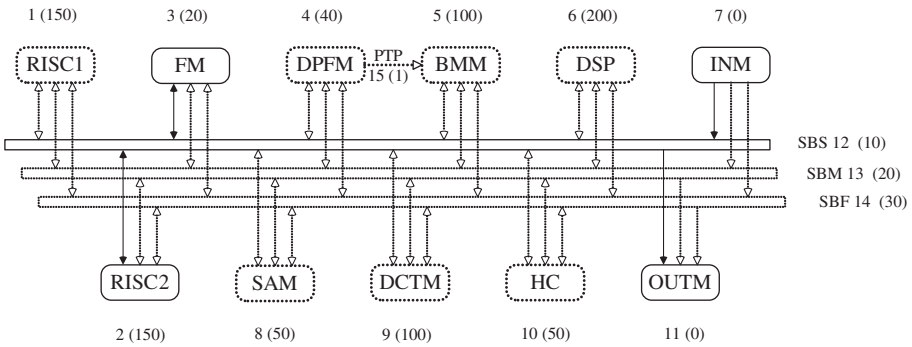

Fig. 28. The hardware implementation for solution (180,166).

population size of 100 and a maximum number of 200 generations. The comparative results are shown in Table 4.

From Table 4, it can be seen that both the HPAEDA and SPEA can find the best Pareto set. So, it can be concluded that both the HPAEDA and SPEA outperform the existing single-objective EA and the RTS + Pareto ranking method. The resulted hardware implementations and the Gantt charts corresponding to the Pareto solutions are shown in Figs. 18–29. Compared to the SPEA, although the HPAEDA yields the same performance, it has some merits. For the SPEA, it needs to design the specific search operators for the SLSP since the problem has a lot of constraints. Different from the SPEA, the HPAEDA is specially designed to solve the project scheduling problem, and it can be used conveniently to solve the SLSP by modeling the problem as the MOMIRCPSP_MS_RI. In addition, the existing theories and methodologies for the project scheduling can also be generalized and employed in the HPAEDA to solve the SLSP. In short, this work provides a new modeling methodology for the SLSP, and it also enriches the solution tool-kit to solve the SLSP in the sense of multi-objective optimization effectively. As for the other applications of the HPAEDA, first it can be used to solve various multi-objective resource constrained project scheduling problems even with different objectives, and second it can be used to solve other real problems which can be modeled as the project scheduling problems.

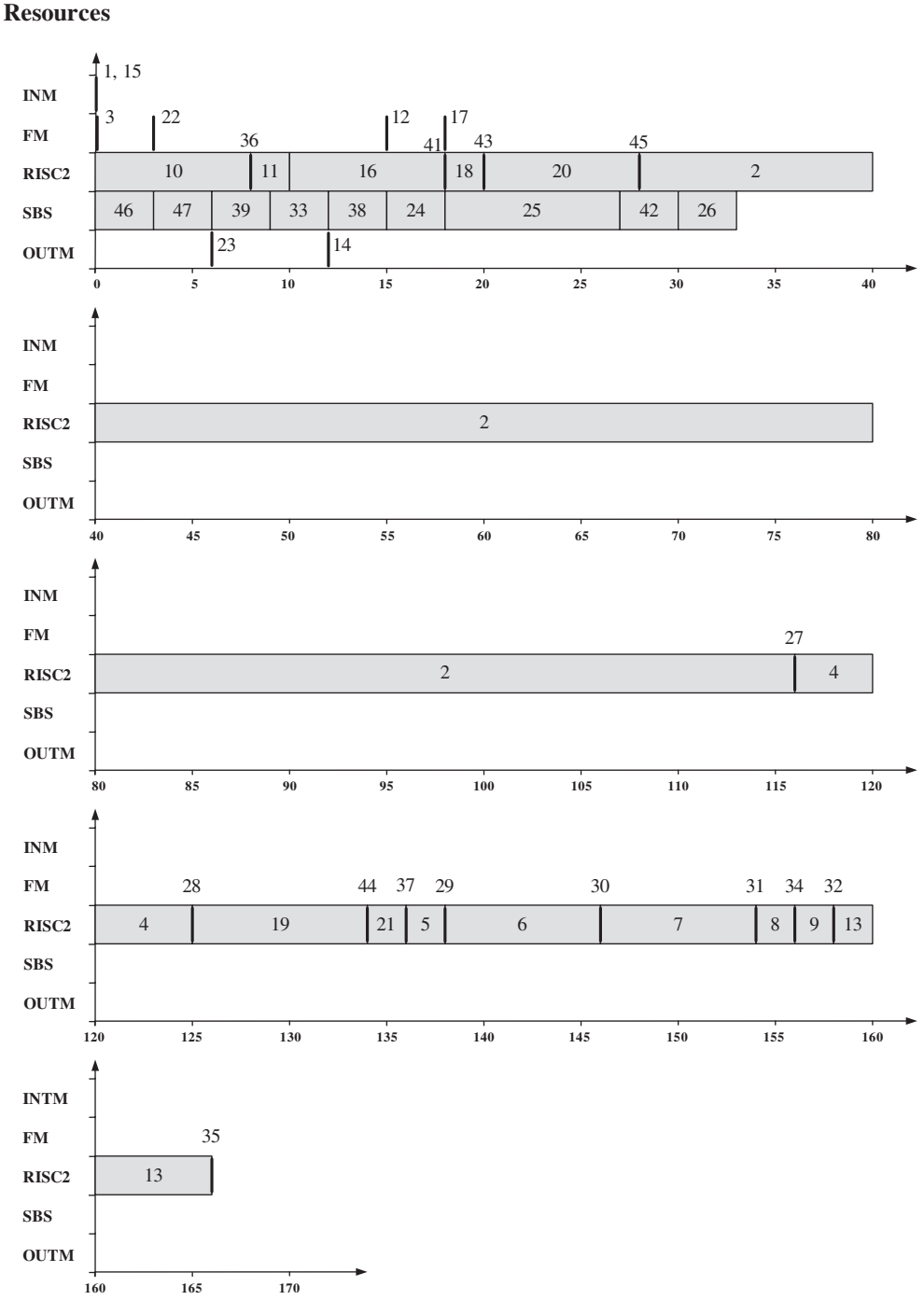

Fig. 29. The Gantt chart for solution (180,166).

7. Conclusion

The main contributions of this paper can be summarized as follows: first, the SLSP is modeled as a multi-objective mode-identity resource-constrained project scheduling problem with makespan and resource investment criteria; second, a hybrid Pareto-archived estimation of distribution algorithm is proposed to solve the problem. This is the first reported work to solve the SLSP in the context of project scheduling modeling, and this is also the first reported work to solve the SLSP by using the multi-objective estimation of distribution algorithm. The modeling methodology and the proposed HPAEDA are tested by the example of a video codec design based on the H.261 image compression standard. Simulation results and comparisons demonstrated the effectiveness of the modeling methodology and solution algorithm. The modeling methodology provides a new understanding to the SLSP, and the HPAEDA provides a new solution algorithm to the SLSP in the sense of multi-objective optimization. This research work could enrich both the modeling methodology and solution tool-kit for the SLSP. Further work could focus on further exploring the problem-specific characteristics, such as the architectural partitioning, to extend the modeling methodology and to improve the algorithm. It is also interesting to develop some adaptive algorithms for the problems with different scales and to apply the algorithm to solve some related real engineering problems that can be modeled as the project scheduling problems.

Acknowledgments

This research is partially supported by the National Key Basic Research and Development Program of China (No. 2013CB329503), National Science Foundation of China (Nos. 61174189, 61025018), Doctoral Program Foundation of Institutions of Higher Education of China (No. 20100002110014), and National Science and Technology Major Project of China (No. 2011ZX02504-008).

References

Aickelin, U., Burke, E. K., & Li, J. (2007). An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. *Journal of Operational Research Society, 58*, 1574–1585.

Armañanza, R., Saeys, Y., Inza, I., García-Torres, M., Bielza, C., van de Peer, Y., et al. (2011). Peakbin selection in mass spectrometry data using a consensus approach with estimation of distribution algorithms. *IEEE-ACM Transactions on Computational Biology and Bioinformatics, 8*, 760–774.

Ballestín, F. (2007). When it is worthwhile to work with the stochastic RCPSP? *Journal of Scheduling, 10*, 153–166.

Ballestín, F., & Blanco, R. (2011). Theoretical and practical fundamentals for multi-objective optimisation in resource-constrained project scheduling problems. *Computers & Operations Research, 38*, 51–62.

Blickle, T. (1996). Theory of evolutionary algorithms and application to system synthesis. Dissertation Techn. Wiss ETH, No. 11894, Zurich.

Blickle, T., Teich, J., & Thiele, L. (1998). System-level synthesis using evolutionary algorithms. *Design Automation for Embedded Systems, 3*, 23–58.

Bovik, A. C. (2005). *Handbook of image and video processing*. Academic Press.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research, 112*, 3–41.

Chen, W. N., & Zhang, J. (2012). Ant colony optimization for software project scheduling and staffing with an event-based scheduler. *IEEE Transactions on Software Engineering, 39*, 1–17.

Demeulemeester, E. L., & Herroelen, W. (2002). *Project scheduling: A research handbook*. Dordrecht: Kluwer Academic.

Dömer, R., Gerstlauer, A., Peng, J., Shin, D., Cai, L., Yu, H., et al. (2008). System-on-chip environment: A SpecC-based framework for heterogeneous MPSoC design. *EURASIP Journal on Embedded Systems, 3*, 1–13.

Fan, Z., Wang, J., Achiche, S., Goodman, E., & Rosenberg, R. (2008). Structured synthesis of MEMS using evolutionary approaches. *Applied Soft Computing, 8*, 579–589.

Gerstlauer, A., Haubelt, C., Pimentel, A. D., Stefanov, T. P., Gajski, D. D., & Teich, J. (2009). Electronic system-level synthesis methodologies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 28*, 1517–1530.

Hanzálek, Z., & Jurcík, P. (2010). Energy efficient scheduling for cluster-tree wireless sensor networks with time-bounded data flows: Application to IEEE 802.15.4/ZigBee. *IEEE Transactions on Industrial Informatics, 6*, 438–450.

Harik, G. R. (1995). Finding multimodal solutions using restricted tournament selection. In *Proceedings of ICGA* (pp. 24–31).

Keinert, J., Schlichter, T., Falk, J., Gladigau, J., Haubelt, C., Teich, J., et al. (2009). SystemCoDesigner-an automatic ESL synthesis approach by design space exploration and behavioral synthesis for streaming applications. *ACM Transactions on Design Automation of Electronic Systems, 14*, 1–23.

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research, 90*, 320–333.

Kuster, J., Jannach, D., & Friedrich, G. (2009). Extending the RCPSP for modeling and solving disruption management problems. *Applied Intelligence, 31*, 234–253.

Larrañaga, P., & Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Netherlands: Springer.

Li, K. Y., & Willis, R. J. (1992). An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research, 56*, 370–379.

Liu, S. S., & Wang, C. J. (2012). Optimizing linear project scheduling with multi-skilled crews. *Automation in Construction, 24*, 16–23.

Lorenzoni, L. L., Ahonen, H., & de Alvarenga, A. G. (2006). A multi-mode resource-constrained scheduling problem in the context of port operations. *Computers & Industrial Engineering, 50*, 55–65.

Lozano, J. A., Larrañaga, P., Inza, I., & Bengoetxea, E. (2006). *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*. New York: Springer-Verlag.

Mann, Z. A., & Orbán, A. (2003). Optimization problems in system-level synthesis. In *Proceedings of Hungarian–Japanese symposium on discrete mathematics and its applications* (pp. 1–10).

Mika, M., Waligora, G., & Weglarz, J. (2008). Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research, 187*, 1238–1250.

Nagaraj Shenoy, U., Banerjee, P., & Choudhary, A. (2000). A system-level synthesis algorithm with guaranteed solution quality. In *Proceedings of EURO-DAT* (pp. 417–424).

Niemann, R., & Marwedel, P. (1997). An algorithm for hardware/software partitioning using mixed integer linear programming. *Design Automation for Embedded Systems, 2*, 165–193.

Nikolov, H., Stefanov, T., & Deprettere, E. (2008). Systematic and automated multiprocessor system design, programming, and implementation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27*, 542–555.

Nikolov, H., Thompson, M., Stefanov, T., Pimentel, A. D., Polstra, S., Bose, R., Zissulescu, C., & Deprettere, E. F. (2008). Daedalus: Toward composable multimedia MP-SoC design. In *Proceedings of DAC* (pp. 574–579).

Pan, Q. K., Wang, L., Mao, K., Zhao, J. H., & Zhang, M. (2013). An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process. *IEEE Transactions on Automation Science and Engineering, 10*, 307–322.

Quintanilla, S., Pérez, Á., Lino, P., & Valls, V. (2012). Time and work generalised precedence relationships in project scheduling with pre-emption: An application to the management of service centres. *European Journal of Operational Research, 219*, 59–72.

Rosado-Munoz, A., Bataller-Mompeán, M., Soria-Olivas, E., Scarante, C., & Guerrero-Martínez, J. F. (2011). FPGA implementation of an adaptive filter robust to impulsive noise: Two approaches. *IEEE Transactions on Industrial Electronics, 58*, 860–870.

Sagarna, R., & Lozano, J. A. (2005). On the performance of estimation of distribution algorithms applied to software testing. *Applied Artificial Intelligence, 19*, 457–489.

Schwiegershausen, M., Kropp, H., & Pirsch, P. (1996). A system level HW/SW partitioning and optimization tool. In *Proceedings of DAC* (pp. 120–125).

Simionescu, P. A., Beale, D., & Dozier, G. V. (2006). Teeth-number synthesis of a multispeed planetary transmission using an estimation of distribution algorithm. *Journal of Mechanic Design, 128*, 108–115.

Su, W., & Chow, M. (2012). Performance evaluation of an EDA-based large-scale plug-in hybrid electric vehicle charging algorithm. *IEEE Transactions on Smart Grid, 3*, 308–315.

Teich, J. (2000). Embedded system synthesis and optimization. In *Proceedings of SDA* (pp. 9–22).

Wang, L., & Fang, C. (2011). An effective shuffled frog-leaping algorithm for multi-mode resource-constrained project scheduling problem. *Information Sciences, 181*, 4804–4822.

Wang, L., & Li, L. P. (2011). Fixed-structure H^{∞} controller synthesis based on differential evolution with level comparison. *IEEE Transactions on Evolutionary Computation, 15*, 120–129.

Wang, L., & Li, L. P. (2013). An effective differential harmony search algorithm for solving non-convex economic load dispatch problems. *International Journal of Electrical Power and Energy Systems, 44*, 832–843.

Wang, L., Wang, S. Y., & Xu, Y. (2012). An effective hybrid EDA-based algorithm for solving multidimensional knapsack problem. *Expert Systems with Applications, 39*, 5593–5599.

Wang, L., Wang, S. Y., Xu, Y., Zhou, G., & Liu, M. (2012). A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. *Computers & Industrial Engineering, 62*, 917–926.

Zhao, J. Q., & Wang, L. (2011). Center based genetic algorithm and its application to the stiffness equivalence of the aircraft wing. *Expert Systems with Applications, 38*, 6254–6261.

Zhao, J. Q., Wang, L., Zeng, P., & Fan, W. H. (2012). An effective hybrid genetic algorithm with flexible allowance technique for constrained engineering design optimization. *Expert Systems with Applications, 39*, 6041–6051.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*, 257–271.