2006 CCRTS
THE STATE OF THE ART AND THE STATE OF PRACTICE

Title of Paper: A Flexible Distributed Scheduling Scheme for Dynamic ESG Environments

Student Paper Submission
(Suggested Track: Modeling and Simulation)

Feili Yu[*1]
E-mail: yu02001@engr.uconn.edu

Sui Ruan[*1]
E-mail: sruan@aptima.com

Meirina Candra[*1]
E-mail: meirina@engr.uconn.edu

David Kleinman[2]
E-mail: kleinman@nps.navy.mil

Krishna R. Pattipati[3]
University of Connecticut,
Dept. Of Electrical and Computer Engineering
371 Fairfield Road, Unit 1157
Storrs, CT 06269-1157
Fax: 860-486-5585
Phone: 860-486-2890
E-mail: krishna@engr.uconn.edu

This work is supported by the Office of Naval Research under Contract #N00014-00-1-0101 and #N00014-06-1-080

1. Electrical and Computer Engineering Department, University of Connecticut, Storrs, CT 06269-2155, USA.
2. C4I Academic Group, Naval Post-graduate School, 589 Dyer Road, Monterey, CA 93943, USA.
3. Correspondence: krishna@engr.uconn.edu

*. Ph.D. Student

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2006		**00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Flexible Distributed Scheduling Scheme for Dynamic ESG Environments	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Connecticut,Department of Electrical and Computer Engineering,371 Fairfield Road Unit 1157,Storrs,CT,06269-1157**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**36**	

A Flexible Distributed Scheduling Scheme for Dynamic ESG Environments

Feili Yu, Sui Ruan, Meirina Candra, David Kleinman, and Krishna Pattipati

***Abstract*— Based on the holonic C^2 organizational control architecture (OCA) that models a C^2 organization as an integration of multi-level, de-centralized decision making networks, we present a holonic multi-objective evolutionary algorithm ($MOEA$) that produces robust and flexible distributed schedules within a dynamic ESG mission environment, such as asset break down, appearance of new events, node failures, etc. The lower level units generate multiple local schedules based on local resources, constraints, and interests (objectives). These local schedules correspond to a schedule pool, from which the Operational Unit can assemble a set of ranked $L-Neighboring$ global schedules according to global objectives, and the actual schedule can shift among different stages of alternative schedules in order to adapt to environmental changes. Global feasibility is ensured at the upper level operational unit, while local autonomies are maintained among lower tactical level units due to the characteristics of the proposed holonic organizational control architecture (OCA). The advantage of this scheduling scheme is that it generates multiple neighboring candidate schedules, which avoids the costly replanning process and also minimizes the adaptation cost.**

Keywords: command and control (C^2), decision maker (DM), expeditionary strike group (ESG), multi-objective evolutionary algorithm ($MOEA$), operational level control (OLC), organizational control architecture (OCA), Pareto-optimal solution, tactical level control (TLC), tactical level control unit (TU), task completion accuracy (TCA).

I. Introduction

A. Motivation

An Expeditionary Strike Group (ESG) is a new US Navy task force that integrates amphibious warships and marines with a tomahawk missile-capable cruiser and destroyer, a frigate and fast-attack submarine. ESGs enhance naval expeditionary force survivability by transforming a previously vulnerable, yet highly valuable, asset into a more combat credible force package [1]. The current planning/scheduling schemes may have worked well in times when missions are deterministic, resources are ample, and operations are conducted within limited areas, i.e., a static environment. As we look towards a dynamic environment of the future, planning/scheduling scheme should be robust and adaptive. The design of future ESG planning/scheduling scheme need to consider the following issues:

1) *Scarcity of Resources:* In general, decision-makers are provided with limited resources with which to accomplish their objectives. The capability of resources may degrade after each operation (i. e., require maintenance) and some assets may break down due to incidents. This makes the distribution of resources among ESG units and the assignment of these resources that enables task processing an important element of ESG planning/scheduling. For example, instead of statically assigning assets to a certain decision maker, one can establish an agile planning/scheduling scheme so that all entities in the ESG can share the limited resources dynamically to meet the changing mission needs. This is also one way to address the 'revolving door' problem associated with assets.
2) *Concurrent Operations:* Conducting multiple concurrent operations that span the entire spectrum of missions is an integral part of an ESG. A complicating factor is that ESGs are being tasked with many unanticipated operations that are not planned for or scheduled a priori. The requirement to perform these concurrent operations is creating situations where com-

manders are performing novel tasks, using new processes and, in some cases, using assets in new ways. An extremely adaptive and dynamic planning/scheduling scheme is needed to keep track of all these concurrent operations.

3) *Dispersed Operations:* ESG's assets are likely to be dispersed geographically to some location outside the immediate battle-space. The planning/scheduling scheme of distributing information, resources, and activities among ESG units must be set up to achieve timely mission processing, while efficiently utilizing the assets. At the same time, ESG is required to detach assets to support other commanders, and to assume control of additional assets from coalition partners. This also requires an agile planning/scheduling scheme.
4) *Local Priorities versus Global Mission Objectives:* The new planning/scheduling scheme should be able to resolve conflicts among local priorities and over-all mission objectives. One abstract solution is to design a planning/scheduling scheme such that global control can be coupled with local priorities. The satisfaction of global objectives can either be achieved by exchange of meta-level information, or by command via a higher level decision maker.

B. Multi-Objective Optimization and Evolutionary Algorithms ($MOEAs$)

In a single-objective optimization problem, the primary focus is on finding a global optimum, representing the best objective function value. However, real world optimization problems often involve more than one objective, which may be conflicting with each other, and, thus, no global optimum can be found. Evolutionary algorithms are well-suited to multi-objective optimization problems, because they are based on biological processes that are inherently multi-objective [7]. $MOEA$s have been extensively studied during the past decade. These are categorized into three distinct classes depending on fitness functions and selection schemes. They are *criterion selection, aggregation selection,* and *Pareto selection*. Criterion selection approaches, such as vector evaluated genetic algorithm ($VEGA$) [8], produce the offsprings from the sub-populations that are selected from a global population by focusing on each objective. Aggregation selection methods, represented by the work in [9], use a weighted sum of fitness values from different objectives. Pareto selection approaches are the most popular in the field of $MOEA$s. They are represented by the following: Pareto Ranking [29], Niched Pareto Genetic Algorithm ($NPGA$) [11], Non-dominated Sorting Genetic Algorithm ($NSGA$) [10], $NSGAII$ [12], Strength Pareto Evolutionary Algorithm ($SPEA$) [13], and Pareto Archive Evolutionary Strategy ($PAES$) [14]. For a comprehensive survey of $MOEA$s, the reader is referred to [15] and [16].

Pareto-based $MOEA$s focus on finding a set of promising solutions (*Pareto optima*), from which a solution can be chosen, if it satisfies certain requirements. A solution, $\bar{s_1}$, is said to be a *Pareto optimum*, if it is not *dominated* by any other feasible solution, say $\bar{s_2}$, that is, the solution $\bar{s_1}$ is not worse than $\bar{s_2}$ in all objectives and better than $\bar{s_2}$ in at least one objective. In other words, solution $\bar{s_1}$ is *Pareto non-dominated* by solution $\bar{s_2}$. However, searching for *Pareto optimum* usually gives more than one solution, which comprise a boundary that is called *Pareto front*. The basic idea in $MOEA$s is to find the set of chromosomes comprising a *Pareto front* in the population that are *Pareto non-dominated* by the rest of the population. A bi-objective minimization problem, illustrated in Fig. 1, shows that individuals 1, 2, 3 are non-dominated points, which comprise

the *Pareto front*. The individual S_k (7) is dominated by 2, 4, 5 and it dominates 9, 10. Several techniques, widely applied in the field of $MOEA$s, that are different from those in single objective optimization algorithms, are the following:

- *Fitness assignment* is an important component that guides multi-dimensional search toward the *Pareto front*. Pareto ranking, domination counting, and non-dominated sorting are widely used in $MOEA$s. Although variations do exist, the common element of these methods is that they are all based on domination or non-domination counting, which introduces the issue of computational efficiency. The work in [6] proposed methodologies to reduce computational demands for some published $MOEA$s.
- *Elitism* is an archive of the Pareto optimal solutions found so far during search. Zitzler *et al.* [17] found that *elitism* is an important factor in improving evolutionary multi-objective optimization. However, how best to utilize *elitism* (e. g., how *elitism* participates in selecting the next generation) is still an open research topic.
- *Niching and fitness sharing* are the two mechanisms that ensure diversity in a population. Between the two solutions with the same Pareto rank, the one that resides in a sparsely populated search-space is preferred. This is called *niching*. The *fitness sharing* usually involves a sharing parameter σ_{share} denoting the largest distance metric, with which a user can decide if any two solutions share each other's fitness. Although *Niching and fitness sharing* can maintain sustainable diversity in a population, they introduce higher computational complexity to those $MOEA$s that apply them. Some recent algorithms, such as $NSGA-II$ [12], have partially alleviated the above problems.

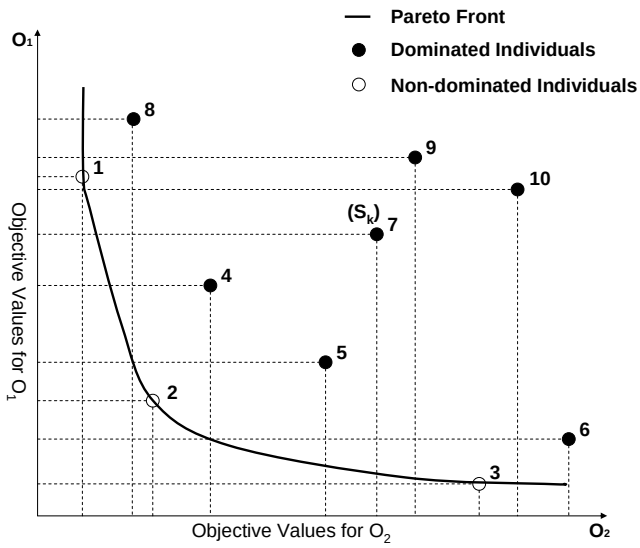


Fig. 1. Example of a bi-objective Pareto fitness domain

C. Flexible and Distributed Scheduling with EAs

In a military battle field, it is vital for commanders to find the right task execution sequence (schedule), and associate the schedule with the right set of assets and decision makers. Furthermore, since the task execution environment is dynamic and uncertain, the proposed schedule should be able to compensate for such changes or uncertainties, i. e., the schedule should be robust and/or adaptive. In an uncertain environment, the objective functions and the constraints of an optimization problem may change over time. Consequently, the optimal solutions may change as well.

One of the ways to react to an environmental change is to restart the optimization process by considering the changes to objective functions, constraints, or optimization parameters. However, this approach is time consuming and a rapid response to an emergent change is crucial in military operations. A speedup re-optimization scheme needs to take advantage of previous optimization information by assuming that the new optimal solution is close to the old one in the search space. For example, in [18] and [19], corresponding genes are inserted into the chromosome when a new task arrives.

Some other EAs use variable mutation rates to shift search area to an area near the new optimal solution after a change occurs, such as those in [20] and [21]. Both of these two types of EAs have a shortcoming, in that, if the amount of changes is large, the EA may not converge due to the frequent discontinuity in the optimization process. Another alternative is the use of previously stored information from past generations to match different environments corresponding to various changes as in [22] and [23]. In recent years, $MOEA$s are used in solving dynamic Pareto optimization problems. Yamasaki [27] proposed a dynamic GA to cope with changing environments. Farina *et.al* [24] formulated dynamic multi-objective optimization problems and proposed several test cases. An Artificial Life ($ALife$) inspired $MOEA$ is proposed in [25] to ensure the approximation of Pareto-optimal front in the presence of unpredictable parameter changes. Bui *et.al* [26] solved single-objective optimization problems using $NSGA-II$ by adding an artificial objective. These works reveal that there is a very close relationship between $MOEA$s and uncertain environments.

In our previous work in [2], we proposed a holonic organizational control architecture, in which resources and controls are distributed among several agents, i.e., several tactical level control units (TUs) and an operational unit. The consequence of distribution is that each unit forms its own objectives and schedules that might conflict with others during the process of scheduling. In this paper, we incorporate multi-objective optimization techniques mentioned above to seek a set of feasible schedules that adapt to environmental disturbances. The lower level units generate multiple local schedules based on local resources, constraints, and interests (objectives). These local schedules correspond to a schedule pool, from which the Operational Unit can assemble a set of ranked $L-Neighboring$ global schedules according to global objectives, and the actual schedule can shift among different stages of alternative schedules in order to adapt to environmental changes. Global feasibility is ensured at the upper level operational unit, while local autonomies are maintained among lower tactical level units due to the characteristics of the proposed holonic organizational control architecture (OCA).

D. Organization and Scope of the paper

The focus of this paper is to propose an agile scheduling scheme to account for mission and organizational arising in dynamic ESG mission environments. In section II, a multi-objective model for flexible holonic scheduling is presented, where the processes of multi-objective problem formulation, co-operative mechanism, and dynamics of the model are discussed. In section III, flexible scheduling solution approaches, employing multi-objective optimization techniques at the operational and tactical levels, are described. An illustrative example and numerical simulation are given in section IV. Finally, the paper concludes with a summary and open topics that need to be explored in the future.

II. Multi-Objective Model for Distributed Holonic Scheduling

The multi-objective model for the C^2 OCA application includes tasks, assets, task precedence constraints, planning and scheduling parameters, multi-level objectives, co-operative mechanism, and dynamics of the model.

A. Tasks

A task, derived from mission decomposition, is an activity that entails the use of relevant resources (provided by assets), and is carried out by one or more decision makers (DMs) to accomplish the mission objectives. In our model, we characterize a

task T_i ($i = 1, ..., I$, where I is the number of tasks) by specifying the following basic attributes:

- Task start time $t_{s,i}$;
- Task processing time $t_{p,i}$;
- Task finish time $t_{f,i}$;
- Geographical location of task $(x_{t,i}, y_{t,i})$;
- Task resource requirement vector $R_{t,i} = [r_{i,1}, \ldots, r_{i,l}, \ldots, r_{i,L}]$, where $r_{i,l}$ is the number of units of resource l ($l = 1, \ldots, L$, where L is the number of resource types) required for successful processing of task T_i.

B. Assets

An asset (or platform) is a physical entity with specified resource capabilities, range of operation, and velocity that is used to process tasks. For each asset $P_j (j = 1, ..., J$, where J is the number of assets), we define the basic attributes as follows:

- Initial geographical location of asset $(x_{p,j}, y_{p,j})$;
- Asset resource capability vector $R_{p,j} = [r_{j,1}, \ldots, r_{j,l}, \ldots, r_{j,L}]$, where $r_{j,l}$ is the number of units of resource l possessed by asset P_j;
- Asset's maximum velocity v_j.

C. Task Precedence Constraints

The Operational Decision maker ($OPDM$) in the Operational Level Control (OLC) architecture devises a plan for the mission that includes the processing of tasks [2]. The task decomposition knowledge is often held by the $OPDM$, but in some cases, it might be necessary to negotiate with other DMs before the criteria for decomposition are established. We define the set of immediate successor tasks of task T_i as I_i^s. Then, the task precedence constraints are given by

$$t_{s,i'} \geq t_{f,i} + t^{timeout}, with T_{i'} \in I_i^s \tag{1}$$

where $t^{timeout}$ is the required 'timeout' between task T_i and its succeeding tasks.

D. Planning and Scheduling Parameters

The planning DM ($PLDM$) in the OLC architecture is responsible for addressing the following planning issues:

- The number of tactical level control units (TUs), M;
- Task to TU assignment, which is given by

$$g_{i,m} = \begin{cases} 1, & \text{if task } T_i \text{ is assigned to } TU\ m; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

- Asset to TU allocation, which is given by

$$q_{j,m} = \begin{cases} 1, & \text{if asset } P_j \text{ is allocated to } TU\ m; \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

We define the design parameter, the optimal or near optimal asset to task scheduling variable, as follows:

$$h_{i,j}(b,c) = \begin{cases} 1, & \text{if asset } P_j \text{ is allocated to task } T_i \\ & \text{during time interval } [b,c]; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

E. Multi-level Objectives, Co-operative Mechanism and Dynamics of the Model

The holonic agent network is a system composed of a population of autonomous DM units, which cooperate with each other to achieve common objectives, while, at the same time, each DM unit pursues its own individual objectives. The objectives can be specified at two levels: Tactical Level Objectives and Operational Level Objectives.

1) Tactical Level Objectives: At the tactical level, each tactical unit (TU) seeks to fulfil the tasks assigned to it with the highest task completion accuracy (TCA). In addition, each TU seeks to reduce its task execution cost (TEC) as much as possible. The expressions for TCA and for TEC are given below.

- *Maximize Task Completion Accuracy (TCA) at TLC*

We adopt the concept of task accuracy from [3]. When all resources required by task T_i are available, the accuracy of task completion equals 100%. However, in realistic applications, where the resources are scarce, an organization may wish to reduce the task execution accuracy in order to achieve better timeliness. In order to accommodate timeliness-accuracy trade-off, the task accuracy for a task T_i executed by $TU\ m$ is defined as

$$A_{i,m} = \left(\frac{1}{\tilde{L}} \sum_{l=1}^{L} a_{i,l} \ \cdot g_{i,m} \right) \tag{5}$$

where $\tilde{L}$ is the number of resource requirements of task T_i over all types, $i.e.,\quad \tilde{L} = \sum_{l=1}^{L} r_{i,l}$, $a_{i,l}$ is the task accuracy for T_i in terms of each resource type l, which identifies the percentage of satisfied resources for the corresponding resource type and is evaluated via $a_{i,l} = \frac{\tilde{r}_{i,l}}{r_{i,l}}$, where $\tilde{r}_{i,l}$ is the number of resources of type l actually used to process task T_i :

$$\tilde{r}_{i,l} = \min \left\{ r_{i,l}, \sum_{P_j \in \underline{P}_m} h_{i,j}(b,c) \cdot r_{j,l} \right\} \tag{6}$$

where $\underline{P}_m$ is the asset set allocated to $TU\ m$, and we assume that the duration of T_i is from time index b to time index c. We can see that the task accuracy of a task T_i, $A_{i,m}$, is the average task accuracy over all $\tilde{L}$ resource types.

The TCA maximization problem is formulated as follows:

$$\max_{h_{i,j}(b,c)} \frac{1}{I_m} \sum_{i=1}^{I_m} A_{i,m} \tag{7}$$

subject to (1). where I_m is the number of tasks assigned to TU_m.

- *Minimize Task Execution Cost (TEC)*

While scheduling the execution of one or more tasks, TU would prefer to send minimum set of assets, if a certain TCA criterion is satisfied. This is due to the following two reasons in our problem: firstly, the more assets are used to execute tasks, the higher the travel and coordination costs would be; secondly, if too many assets are allocated to a task, there will not be enough assets to execute other concurrent tasks. We define the travel (transfer) time of assets when executing task T_i as the Task Execution Cost (TEC), which is given by

$$C_{i,m} = \sum_{P_j \in \underline{P}_m} Tr_{i,j} \cdot h_{i,j}(b,c) \tag{8}$$

where

$$Tr_{i,j} = \frac{\sqrt{(x_{p,j} - x_{t,i})^2 + (y_{p,j} - y_{t,i})^2}}{v_j} \tag{9}$$

The TEC optimization problem can be formalized as

$$\min_{h_{i,j}(b,c)} \sum_{i=1}^{I_m} C_{i,m} \tag{10}$$

subject to (1).

In our model, the two objectives are somewhat conflicting. If task execution cost is reduced, it may result in a decrease in task completion accuracy. On the other hand, higher task completion accuracy may increase the task execution cost. The multiple objective optimization problem is to find a set of trade-offs (Pareto front) between these two objectives.

2) Co-operative Mechanism: Co-operative mechanism provides a means to communicate and coordinate among TUs when executing a task. Information and commands are exchanged through negotiation protocols. If a TU cannot complete a task assigned to it using local resources/expertise with a certain task completion accuracy (TCA), it may find other willing TUs (coordinators) with the necessary resources/expertise to coordinate on the task. The schedule of certain assets executing a common task can be negotiated among different TUs. The co-operation is typically achieved via a constraint satisfaction mechanism, that is, to find a set of

assets that satisfy the timing constraints of the task. The procedures for this mechanism are *(i) task announcement; (ii) contracting;* and *(iii) coordinated processing*.

- *Task Announcement*
  The set of tasks assigned to TU_m that have not satisfied their TCA requirements after TLC scheduling are given by

$$\hat{T_m} = \{T_i | A_{i,m} < \alpha, i \in I_m\} \tag{11}$$

  where α is a constant denoting the TCA threshold. The TU_m requires coordination with other units for processing the tasks in $\hat{T_m}$.
- *Contracting*
  The willing TUs are those TUs that (a) possess the required resources/expertise; and (b) own assets that satisfy the timing restrictions on the coordinated tasks. For a task $T_i \in \hat{T_m}$, the candidate set of assets that can satisfy the requirement (a) is given by

$$\begin{aligned}\hat{P} = \{P_j | r_{j,l} \cdot r_{i,l} > 0;\ l = 1, \ldots, L;\\ r_{j,l} \in R_{p,j};\ r_{i,l} \in R_{t,i}\}\end{aligned} \tag{12}$$

  The requirement (b) is introduced in (14) as a constraint. The objective for $contracting$ is given by

$$\max_{h_{i,j}(b,c)} \frac{1}{|\hat{T_m}|} \sum_{i=1}^{|\hat{T_m}|} A'_{i,m}; T_i \in \hat{T_m}, P_j \in \hat{P} \tag{13}$$

  subject to

$$\begin{aligned} h_{j,i}(b,c) = 0 \quad & T_i \in \hat{T_m},\\ b = t_{s,i} - Tr_{i,j}, \quad & P_j \in \hat{P},\\ c = t_{f,i} + Tr_{i,j}, & \end{aligned} \tag{14}$$

  where $A'_{i,m}$ is the TCA of T_i requiring more than one TU, and $Tr_{i,j}$ is the transfer time of asset j assigned to task T_i, which is given by (9).
- *Coordinated Processing*
  Since some tasks require more than one TU to process, we redefine the task accuracy for task T_i as

$$A_i = \begin{cases} A_{i,m}, & if \quad T_i \notin \hat{T_m};\\ A'_{i,m}, & if \quad T_i \in \hat{T_m}. \end{cases} \tag{15}$$

3) Operational Level Objectives: At the operational level, the DMs need to integrate multiple distributed schedules (denoted as S) from each TU to achieve the following two objectives: (a) complete the mission as rapidly as possible; (b) complete the mission with maximum degree of accomplishment. The mission accomplishment is modeled by the overall task completion accuracy, and the mission duration is represented by the makespan of the mission, respectively. The two objectives are formalized as follows:

- *Maximize Task Completion Accuracy (TCA)*
  The overall task completion accuracy at the operational level (TCA) is the mean value of the task completion accuracies. The resulting TCA optimization problem is given by

$$\max_S A = \max_S \frac{1}{I} \sum_{i=1}^{I} A_i \tag{16}$$

  subject to (1).
- *Minimize the Makespan of Mission Execution*
  We assume that the mission start time (t_{Start}) is 0 and mission end time is t_{End}. Minimization of the makespan is equivalent to minimizing the t_{End}. Since tasks are parallel and are distributed among TUs, the makespan is calculated by assembling the schedule from each TU. However, these distributed schedules may conflict with each other and violate the task precedence constraints. Therefore, a global de-confliction procedure will be followed before the makespan can be calculated. The problem is thus given by

$$\min_S t_{End} \tag{17}$$

  subject to (1).

4) Dynamics of the model: To make this model more realistic, we enhance it by adding two dynamic components: (a) *dynamic asset capability*, and (b) *dynamic task processing time*. The latter is a function of task accuracy. We assume that the asset resource capability can be restored after the asset returns to its base after task execution; however, it will not reach its full capability for the next operation because of the weariness of crew and "wear out" of equipment. Thus, for the renewed resource type l, we have

$$r_{j,l}^{renewed} = r_{j,l} \cdot (1 - p_l \cdot \frac{\tilde{r}_{j,l}}{r_{j,l}}) \tag{18}$$

where p_l is the coefficient of weariness for resource type l, $r_{j,l}$ is the full capability of resource type l, and $\tilde{r}_{j,l}$ is the actual number of resource type l used to process previous tasks. For example, if the resource of type l is totally used up, then the renewed resource capability of this resource type is $r_{j,l} \cdot (1 - p_l)$.

We define the *dynamic task processing time* for task T_i as

$$t_{p,i} = \frac{1}{n_i^{\rho} \cdot c_{p,i}^{\beta}} \cdot (e^{A_i^{\gamma}} - 1) \tag{19}$$

where $n_i = \sum_j h_{i,j}(b, c)$ is the number of assets allocated to process task T_i during the time interval $[b, c]$; $c_{p,i}$ is the asset capability for task T_i; ρ, β, and γ are constants. The relationship among task processing time, TCA, and the asset capability is illustrated in Fig. 2, and the relationship among task processing time, TCA, and the number of coordinating assets is shown in Fig. 3.

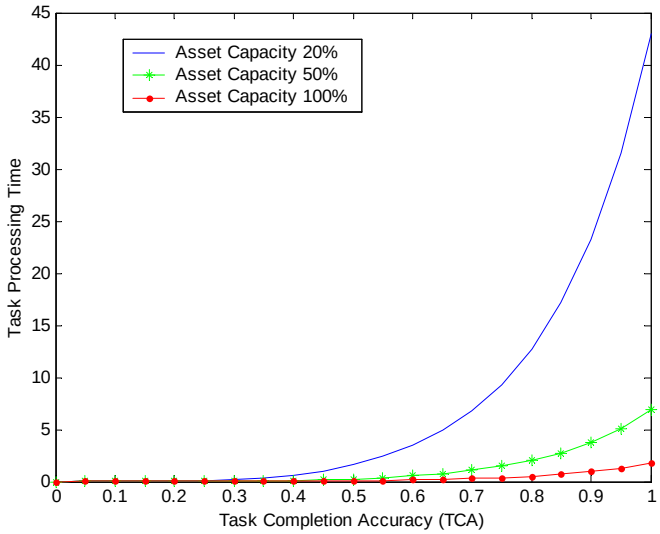


Fig. 2. Task processing time as a function of TCA for assets with different capabilities

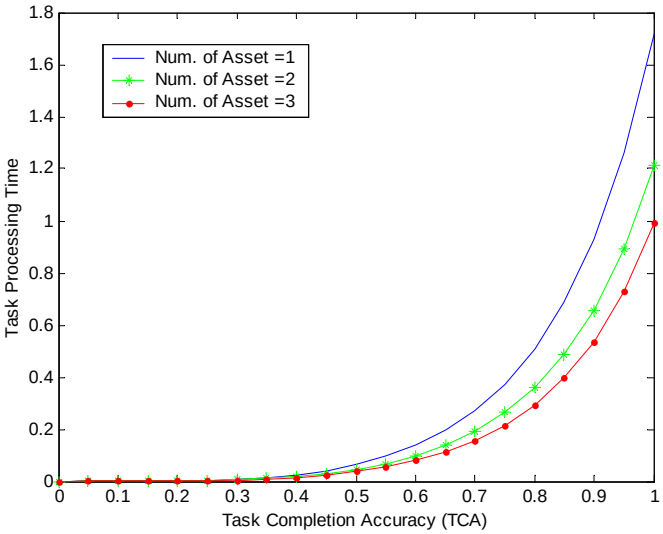


Fig. 3. Task processing time as a function of TCA for different number of coordinating assets

III. Solution Methodology for Flexible Distributed Holonic Scheduling

A holonic scheduling approach differs from a conventional one primarily in terms of the distribution of computation and decision making functions, and the interactive (and largely co-operative) nature of the communication between the DMs.The holonic scheduling process involves interactive coordination and communication among DMs from both the OLC and the TLC levels, i.e. the $COODM$ in the OLC architecture and the $SCDM$ in each TU [2]. Accordingly, the holonic scheduling is comprised of five elements: *central plan decomposition, tactical level scheduling, co-operative negotiation, operational level schedule building,* and *interactive scheduling*.

Algorithm for calculating task priority

While $i \neq number\ of\ total\ tasks$, do
  $T_i \rightarrow Priority = 1$
  If T_i has precedent tasks $\{T_k\}$
    $T_i \rightarrow Priority = max\{T_k \rightarrow Priority\} + 1$
    $i = i + 1$
  Else
    $i = i + 1$
  end if
end do;

Fig. 4. Algorithm for calculating task priority

A. Central Plan Decomposition (Step 1)

Central plan decomposition is carried out by the $COODM$ in the OLC architecture. The $COODM$ first decomposes the central plan and the task precedence graph, and then distributes them among the TUs. The task precedence graph is decomposed based on task priority, that is, firstly, it calculates the priority value of each task (the algorithm is shown in Fig. 7); secondly, it orders the tasks according to their priority values (shown in Fig. 11); finally, it distributes the tasks and task priority information among the TUs.

B. Tactical level scheduling (Step 2)

Tactical level scheduling is a distributed process, in which each TU makes the scheduling decisions based on local information, local objectives, and constraints. As mentioned before, the multiple objectives for each TU are: (a) minimizing task execution cost; and (b) maximizing task completion accuracy (TCA). A multi-objective solution approach based on $NSGA-II$ [12] is proposed to solve the scheduling problem. Our improvements to the $NSGA-II$ include a fast sorting scheme, and a Pareto-based scale-independent fitness function. This fitness function is also used in [28]. The procedures are described as follows.

1) Chromosome Representation: The key problem is to find an appropriate assignment of assets to tasks within each TU during a certain time interval, i.e., $h_{i,j}(b,c)$. This is a combinatorial optimization problem. Different assignments may result in different task completion accuracy (TCA) and task execution cost (TEC). Each task is allocated a certain number (may be zero) of candidate assets after a preselection of candidate assets is performed. Based on that, a binary chromosome contains $Q_t = I_m \cdot J_m$ genes, where I_m and J_m are the number of tasks and the number of assets controlled by $TU\ m$, respectively. Each gene represents the assignment status of each asset to each task. If its value is '1', it indicates that the asset has been allocated to this task. For example, for a three-task and three-asset problem, a chromosome taking the form [(1 0 0), (1 0 1), (0 1 0)] indicates that P_1 is allocated to T_1; T_2 will be processed by P_1 and P_3; and P_2 will execute T_3.

2) Fitness Function: For a solution s_i, we define

- p_k, the number of solutions that can be dominated by s_k;
- q_k, the number of solutions that can dominate s_k;
- c_k, the number of total solutions.

The generalized Pareto-based scale-independent fitness function ($GPSIFF$) is given by

$$FITNESS(s_k) = p_k - q_k + c_k \qquad (20)$$

The $GPSIFF$ takes advantage of information of both dominated and non-dominated individuals of solution s_k, which are obtained from all participant individuals in the search space [28]. A bi-objective example, illustrated earlier in Fig. 1, shows that the individual s_k dominates solution 9 and 10 ($p_k = 2$), and s_k is dominated by solution 2, 4, and 5 ($q_k = 3$). Therefore, the fitness value for individual s_k is 2-3+10=9.

3) Fast Non-dominated Sorting: Different from the non-dominated sorting in $NSGA-II$ [12], we apply a pre-sorting scheme, which is tailored for

Algorithm for non-dominated sorting

Step 1: Sort the solutions in ascending order w.r.t one objective, $O_{1,k}$, where $k = 1, \ldots, K$. Denote the sorted set as $\aleph$;

Step 2: For an individual solution $s_k \in \aleph$, set $p_k = |\{s_i | s_j(o_2) < s_k(o_2), s_i \in \aleph, i = 1, \ldots, k-1\}|$; $q_k = |\{s_j | s_j(o_2) > s_k(o_2), s_j \in \aleph, j = k+1, \ldots, K\}|$

Step 3: Repeat *Step 2* until $k == K$

Fig. 5. Algorithm for non-dominated sorting

the bi-objective problem, to accelerate the sorting process. The population of size K is sorted in ascending order with respect to one of the values of the two objectives, i.e., $O_{1,k}$, where $k = 1, \ldots, K$. Within this pre-sorted set, the *Pareto front* is found by sweeping the individual solution one by one with respect to the second objective values, i.e., $O_{2,k}$. It guarantees that if an individual solution $s_k^{'}$ is swept after s_k, then s_k dominates $s_k^{'}$. The pseudo code of this algorithm, shown in Fig. 8, illustrates how the number of domination p_k and the number of non-domination q_k of an individual solution s_k are obtained.

4) Elitism and Nitching: Similar to $NSGA-II$, the new population is produced in such a way that 40% of it is from the old population with the highest fitness values, and the remaining 60% is from the offsprings with the highest fitness values. By doing this, we can maintain both the elitism and diversity in the new population. Another way of maintaining diversity is to use *nitching*, where a crowding distance assignment is calculated for each individual solution by averaging the distances to its neighboring individuals. If several individual solutions share one fitness value, the ones in less crowded area will be picked as new population.

5) Feasibility: The feasibility of the global schedule is ensured if each local schedule is feasible. Due to the different asset to task assignments, some of them may be infeasible, i.e., the completion time of previous task is beyond the start time of the next task. The reason for infeasibility is that the assigned assets to a previous task cannot achieve certain task completion accuracy, so they take more time than the time slot assigned to the task, or, it maybe because a low speed asset is assigned to a task, which is far away from it; the long transfer time causes the delay in task execution. For such infeasible solutions, we assign a very low fitness value (high penalty), so that they will not be selected as part of the new population.

The $MOEA$ running in TU_m produces $N_{s,m}$ ($N_{s,m} \geq 1$) local schedules. $N_{s,m}$ does not have to be the same for different TUs. Some TU_m produce fewer schedules because there may not be enough feasible solutions due to task start time constraints. In this case, the planning DM ($PLDM$) at OLC may need to adjust the initial start time of each task (loosen the time constraints) in order to generate additional local schedules.

C. Co-operative Resource Re-deployment (Step 3)

After *Task Announcement*, tasks whose TCA have not reached certain accuracy are identified (given by (11)) and the candidate assets that can execute the task are also selected (given by (12)). The objective is to maximize the mean task completion accuracy given by (13). This is a single-objective optimization problem, and we employ a GA to find the best task-to-asset assignment.

1) Chromosome encoding: Similar to the chromosome representation in TLC scheduling, a binary chromosome contains $Q_c = |\hat{T_m}| \cdot |\hat{P}|$ genes, where $|\hat{T_m}|$ and $|\hat{P}|$ are the number of the unfinished tasks and the number of candidate assets, respectively.

2) GA Parameters:

- *Fitness function:* The mean task completion accuracy is used as the fitness function, which is given by (13);

- *Operators:* The GA uses *Arithmetic* and *Multi-point* crossover operators and *Multi-non-uniform* mutations. For details on these operators, please refer to our previous work in [4];
- *Selection Strategy:* The new population is comprised of the following three parts: (a) 50% of the new population is from the best 50% of the previous population; (b) 20% of the new population is from crossover; and (c) 30% new population comes from mutations. By doing this, we can keep both elitism and the diversity of the population;
- *Size of initial population and the number of iterations:* The size of initial population is set at $N = 100$, which is sufficient to cover the problem of any size. The number of iterations is set to 50 based on our experimental observation that the GA usually converges within 50 iterations.

The outcome of the co-operative resource redeployment is a coordinating pattern among TUs. The Coordinating DM ($COODM$) at OLC may or may not intervene in this process, depending on whether the global objectives are affected or not.

D. Operational Level Global Schedule Building (Step 4)

Unlike the scheduling process at TLC, the schedule building at OLC would be a multiple schedule selection process to (a) minimize (or maximize) OLC objectives; (b) assemble local schedules into a global schedule; and (c) resolve the conflicts among local schedules and ensure the feasibility of the global schedule.

The $COODM$ at OLC is responsible for building a global schedule by selecting one schedule from each TU to achieve the global objectives: (a) minimize makespan of mission execution; and (b) maximize the overall task completion accuracy. The algorithm finds a set of ranked $L - Neighboring$ schedules. The flexibility is achieved by shifting from one schedule to the other that best fits the current situation. Since the $L-Neighboring$ schedules are neighbors of each other, the cost of adaptivity is small. The $MOEA$ proposed for solving the *Global Schedule Building* problem, except for the encoding and global schedule building process, is similar to the one we used in TLC scheduling. The different parts that are designed for *Global Schedule Building* are described as follows.

1) Chromosome Representation: The index of one of the local schedules in each TU is encoded in the corresponding gene in a chromosome with M genes, where M is the number of TUs. For example, a chromosome [1 3 2 5] indicates that the the first schedule of TU_1, the third schedule of TU_2, the second schedule of TU_3, and the fifth schedule of TU_4 are selected to assemble a global schedule. The $MOEA$ will enumerate the combinations of these indices until a set of $L - Neighboring$ schedules are found.

2) Global Schedule Building Process: Two issues are involved in building a global schedule: (a) to maintain the feasibility of the global schedule; and (b) to converge to a stable schedule state in terms of start time of each task. We use right-shifting to build the global schedule and maintain its feasibility, that is, local TLC schedules are reassembled according to the task graph. The start times of tasks that have precedence constraints will be shifted to a time when all its preceding tasks have been accomplished. This introduces the second issue: how to achieve the stability of the schedule? We employ the following iterative scheme: once the right-shifting is complete, a set of new start times is obtained. We start the entire process with the set of new start times. This iteration is repeated till the task start times do not change. At this stage, the global schedule reaches its stable state. This process is

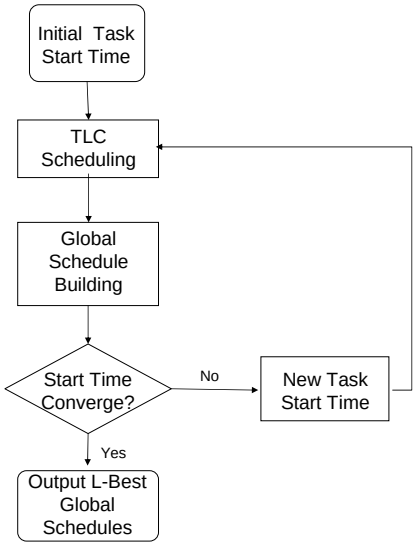


Fig. 6. The global schedule stabilization procedure

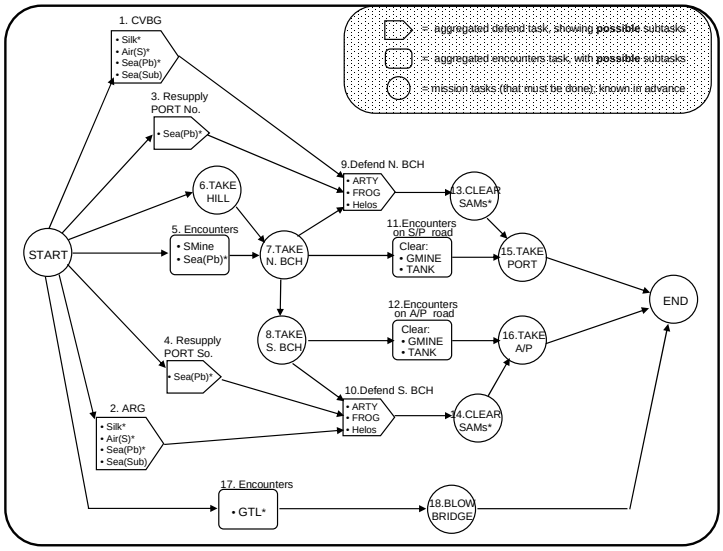


Fig. 8. Task graph for the decomposed mission from the $OPDM$

depicted in Fig. 6.

E. Interactive scheduling (Step 5)

The global schedule can interact with the environment in the following way: each of the $L - Neighboring$ schedules is divided into several stages. At the beginning of each stage, the planning DM ($PLDM$) at OLC evaluates the feasibility of the current schedule based on the reports and data collected by the intelligence DM ($INDM$). Once infeasibility is detected, the $PLDM$ searches among the alternative neighboring schedules at the same stage until a feasible schedule is found. The $PLDM$ then instructs the coordination DM ($COODM$) to issue commands of schedule changes to the lower level TUs. The interactive scheduling procedure is illustrated by Fig. 7.

IV. Illustrative Simulations

A. Mission

A joint group of Navy and Marine Forces is assigned to complete a military mission that includes capturing a seaport and an airport to allow for the introduction of follow-on forces. There are two suitable landing beaches designated "North" and "South", with a road from the North Beach to the seaport, and another road leading from the South Beach to the airport. From intelligence sources, the approximate concentration of the hostile forces is known, and counter-strikes are anticipated.

B. Mission Decomposition and Task Graph

The task graph (shown in Fig. 8) describes the task procedure, the constraints, and the preferences. While the task procedure comprises the task execution logic, the constraints represent task-dependencies, and the preferences specify the task authority structure. In Fig. 9, the asset-resource capability and task-resource requirement matrices are given.

C. Deliberate Planning and Central Plan Decomposition

In the work presented in [4], a nested genetic algorithm was developed to solve the planning problem with the objective of minimizing both the internal and the external workloads of the system. The resulting plan is shown in Fig. 10. The mission plan provides the following information: (a) the optimal number of the TUs for this mission is four ($M = 4$); (b) the task assignment to each TU,

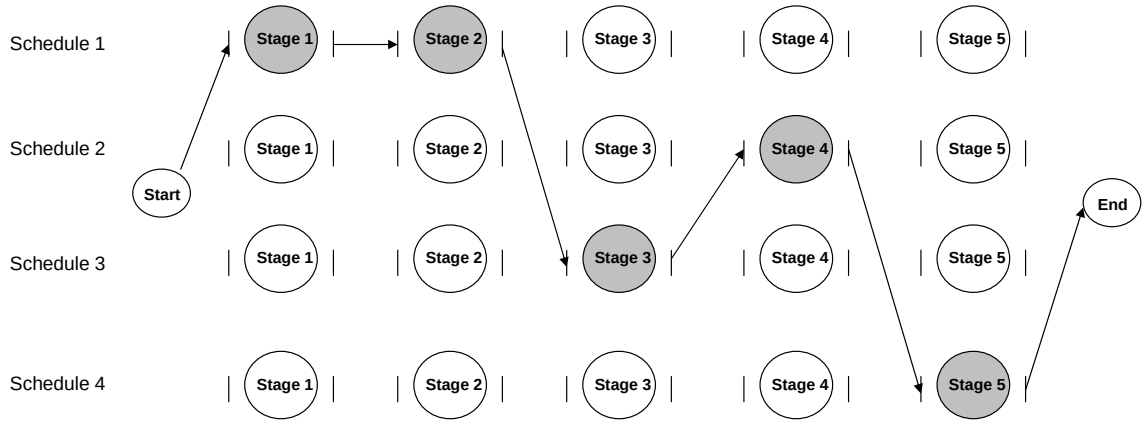


Fig. 7. The interactive scheduling procedure

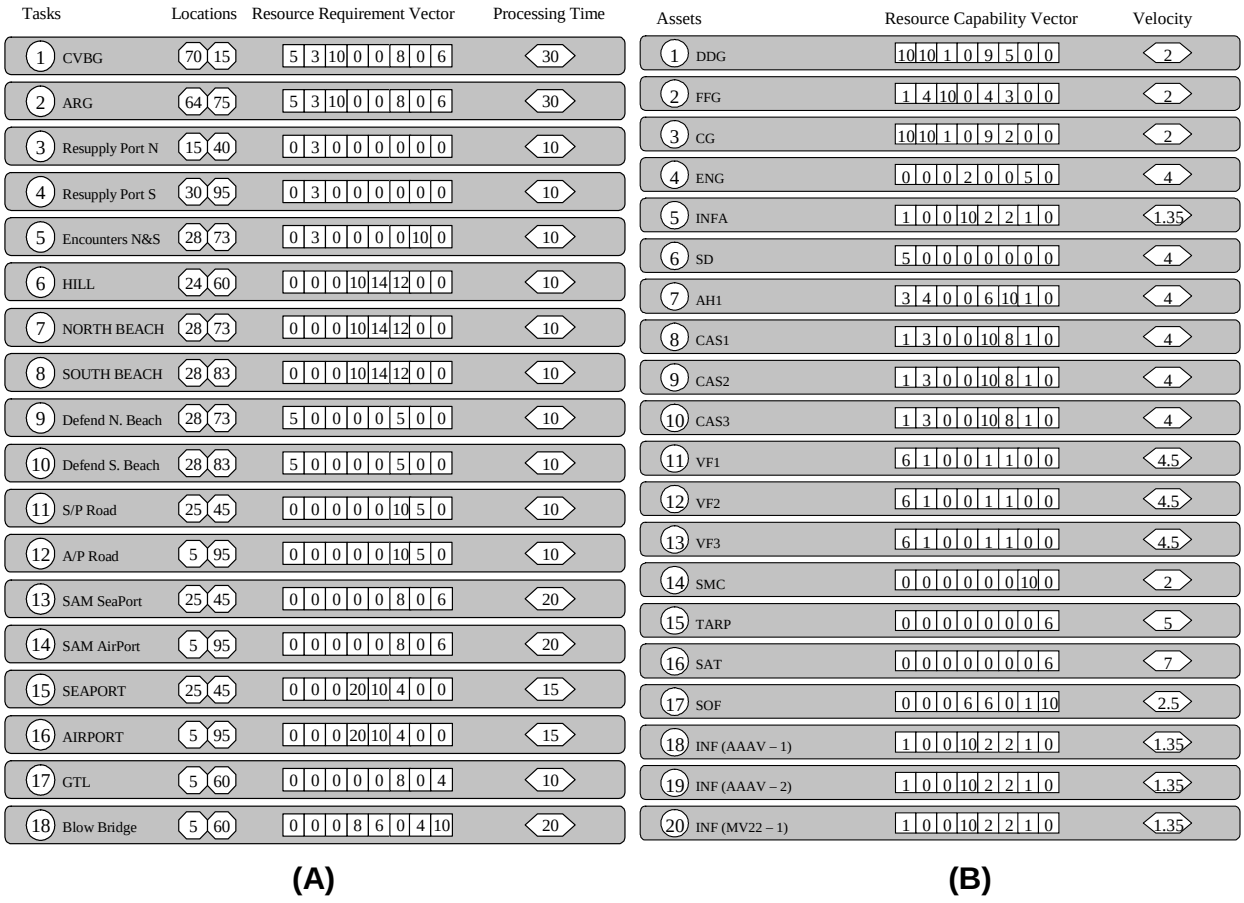


Fig. 9. (A) Task-Resource matrix; (B) Asset-Resource capability matrix

Platforms \ Tasks	3	15	16	18	6	11	12	13	14	17	1	2	4	5	7	8	9	10
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0
17	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0
20	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Fig. 10. The central plan created by the $PLDM$ in OLC architecture

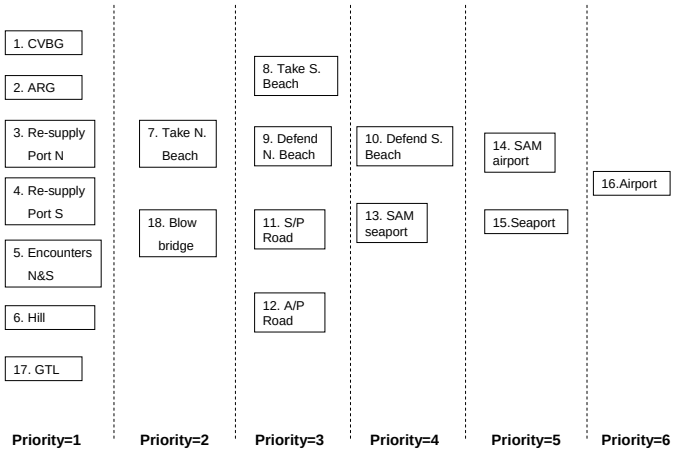


Fig. 11. The task priorities

$\{g_{i,m}\}$; (c) the asset allocation to each TU, $\{q_{j,m}\}$; and (d) the tentative mapping of assets to each task.

The planning DM ($PLDM$) in the OLC architecture decomposes the central plan and the task precedence graph into several sub-plans and sub-task precedence graphs based on the algorithm proposed in Fig. 7, and then distributes them among the TUs. The decomposed task priorities are shown in Fig. 11. The task execution sequence for each TU is illustrated in Fig. 12.

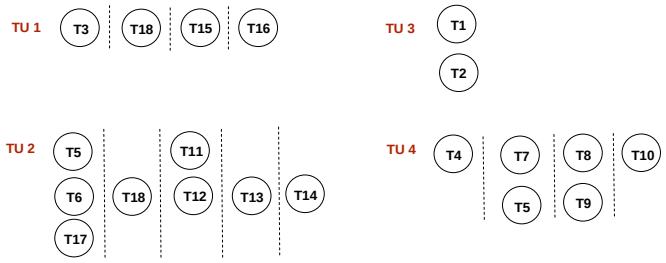


Fig. 12. The distributed task execution sequence for each TU

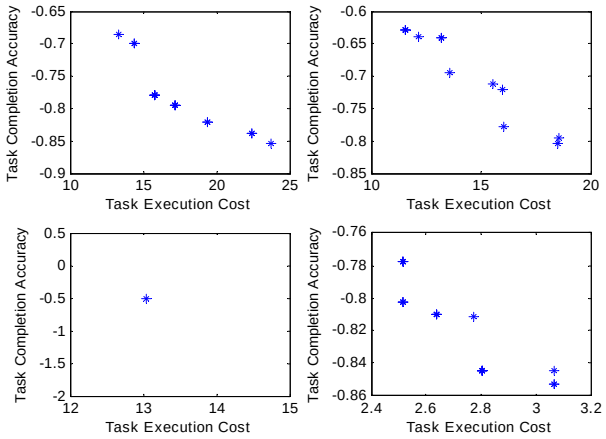


Fig. 13. The Pareto Front found for each TU

D. Tactical Level Scheduling

The $MOEA$ proposed in the previous section is employed to develop tactical level schedules. After it is finished, each TU produces $N_{s,m}(m = 1, \ldots, M)$ Pareto optima. $N_{s,m}$ is different for each TU, which mainly depends on the local asset and the task assignment and also partially on the diversity of the local population. The Pareto front for each TU is shown in Fig. 13. We note that the $N_{s,m}$ for each TU is 7, 9, 1, 7, respectively. Each local Pareto optimum corresponds to a task-asset allocation matrix, therefore, a local Pareto optimal schedule. So, the tactical level scheduling provides a local schedule information pool, from which the operational level global schedule building process can construct global schedules with multiple alternatives.

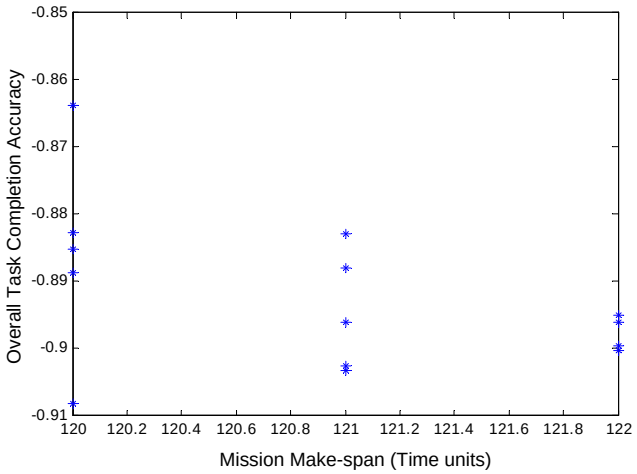


Fig. 14. The $L - Neighboring$ scheduling solutions

TABLE I
THE OBJECTIVE FUNCTION VALUES FOR PARETO OPTIMA

Rank	TCA	Makespan	GPSIFF
1	90.81%	120	106
2	90.32%	121	86
3	90.25%	121	86
4	89.61%	121	83
5	88.86%	120	81
6	88.51%	120	80
7	88.27%	120	78
8	86.37%	120	66
9	88.79%	121	63
10	88.30%	121	61
11	90.03%	122	61
12	89.95%	122	61
13	89.60%	122	60
14	89.50%	122	60

E. Operational Level Schedule Building Process

The operational level schedule building process is responsible for assembling local schedules to form ranked $L - Neighboring$ global schedules, resolving conflicts and achieving OLC objectives. Fig. 14 illustrates the Pareto optima generated by the operational level $MOEA$, which shows that there are 14 Pareto optima present. The objective function values, i.e., the overall TCA, makespan, and $GPSIFF$ for the 14 Pareto optima are listed in Table I. We notice that the range of the TCA is from 88.27% to 90.81%, and the makespan ranges from 120 to 122. This information tells us that even if the mission shifts from the best schedule to the lowest ranked schedule, the performance of the organization will not deteriorate much.

The best schedule is shown in Fig. 15. The Y axis represents the 'Asset ID' and the X axis represents the 'time units'. Each asset is assigned one or more tasks at certain time interval. The length of the gray bar indicates the duration of each task. The schedule can be divided into 5 time stages: 0-29, 30-49, 50-66, 67-88, and 89-122. Between two sequential stages, there are no, or very few interleaving tasks so that the boundary can be identified.

The 14 Pareto optima indicate that there are 14 alternative global schedules that a $COODM$ at the OLC can choose from in order to adapt to a changing environment. The first 10 schedules are plotted in Fig. 16. We notice that the pattern of these schedules is very similar to each other, which means that variations of these schedules are small. This indicates that the cost of adaptivity is small, because the schedules are neighbors of each other. This minimizes the effect of disturbances to the mission operations when the current schedule needs adjustment.

F. Rescheduling under disturbances

The mission commences with the best schedule and shifts to other schedules when disturbances occur because of asset break down or appearance of emergent events. At the beginning of each stage, the planning DM ($PLDM$) at OLC evaluates the status of each asset and decides if a schedule shift is needed. The following two examples illustrate how this process works.

1) Case 1: Asset 10 breaks down: We assume that the mission is proceeding initially according to the best schedule. Asset 10, one of the close air support (CAS) units, encounters strong enemy forces when trying to defend the carrier group (task 1) and suffers heavy casualties during mission stage one. Due to massive loss of its capabilities, this CAS

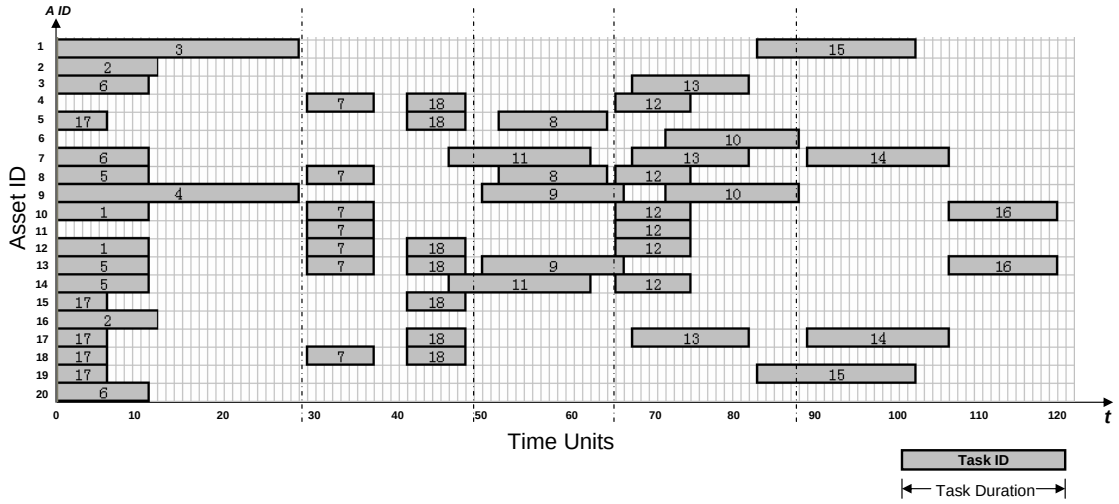


Fig. 15. The best schedule

has to be withdrawn from the battle field. However, according to the first schedule, at time intervals 51-58, 68-76, and 111-120, this CAS unit will execute task 7 (take north beach), task 12 (patrol north part road area) and task 16 (take airport), respectively. At the beginning of the second stage, a schedule shifting decision has to be made due to the absence of asset 10. After a search through the neighboring schedules, an alternative schedule, the third schedule, is chosen. Thus, during the rest of the mission execution, the mission can be executed without asset 10 (see Fig. 17).

2) Case 2: Emerging event: We assume that the current schedule is the first schedule. At time 51, an enemy force unit launches a ground assault toward an airport defended by an allied army. Either an enforcement infantry unit or a special operation force needs to be sent to the airport. This task has the similar resource requirements to task 17. By looking back to stage one, where a similar task has been accomplished, the planning DM ($PLDM$) finds that asset 6, 15, 17, 18 and 19 have been coordinated on a similar task. However, at the current time, asset 6 is going to execute a new task (task 8); therefore, it may not be appropriate to apply this asset. However, the $PLDM$ finds from alternative schedules 4 and 5 that either special operation force (asset 17) or a combination of asset 17 and infantry unit 2 (asset 19) can do the task. The final decision is made to send both assets 17 and 19 to the airport, because this will shorten the task execution time, and has less impact on asset 17's next task execution (task 13). The adjusted schedule is shown in Fig. 18.

V. Conclusions

In this paper, an efficient flexible holonic scheduling scheme applying Multi-Objective Evolutionary Algorithms ($MOEA$) is proposed. Each TU provides multiple Pareto optimal local schedules that satisfy local objectives, while decision makers at the operational level assemble these local schedules, resolve conflicts, and generate a set of ranked $L-Neighboring$ global schedules. When facing environmental disturbances, the upper level DMs can either shift to different stages of alternative schedules or adjust current schedule by learning from the history of the current or alternative schedules. The advantage of this scheduling scheme is that it generates multiple neighboring candidate schedules in one run, which avoids the costly replanning process and also minimizes the adaptation cost.

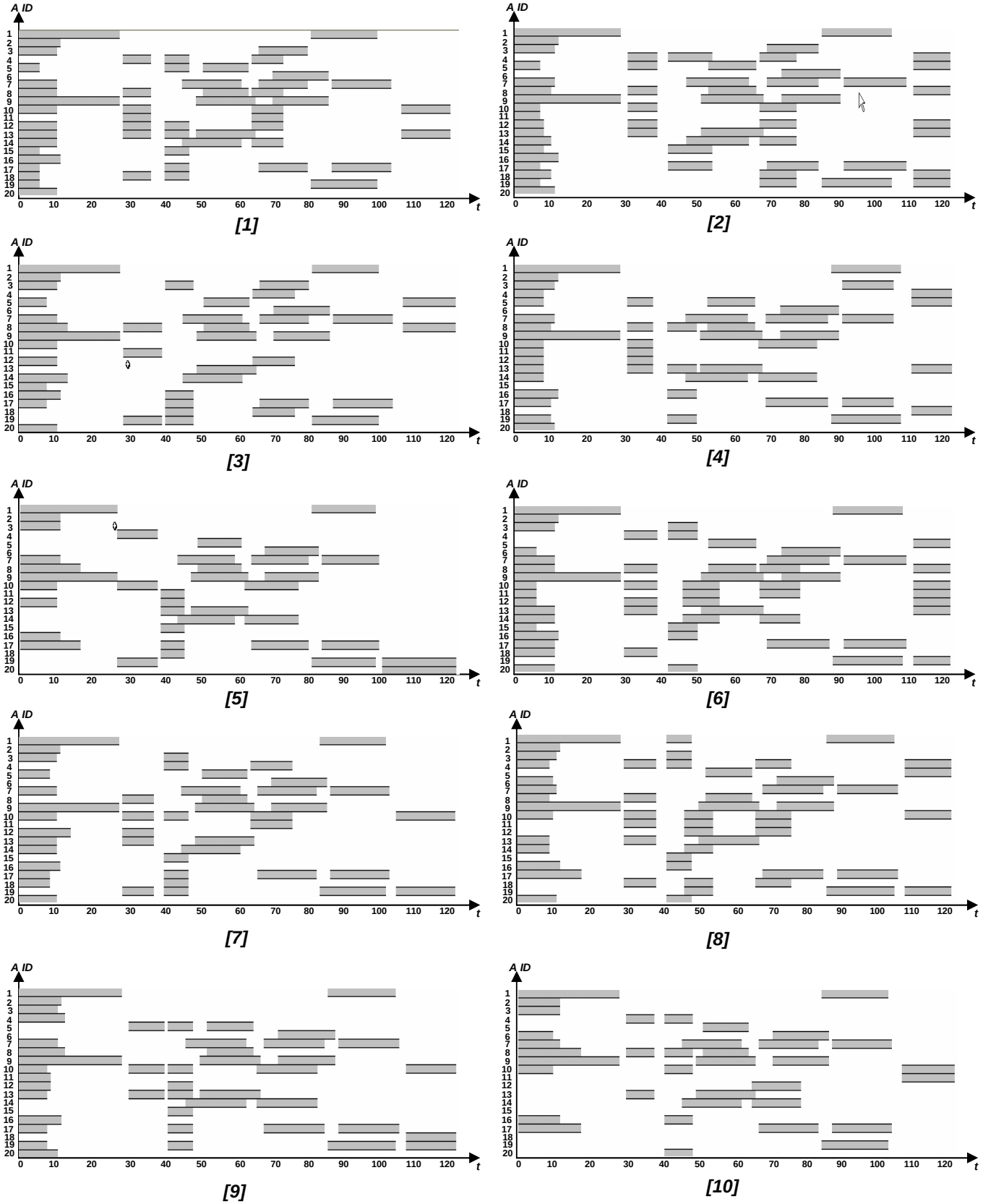


Fig. 16. The ranked $L - Neighboring$ schedules produced by Operational Schedule Building Process

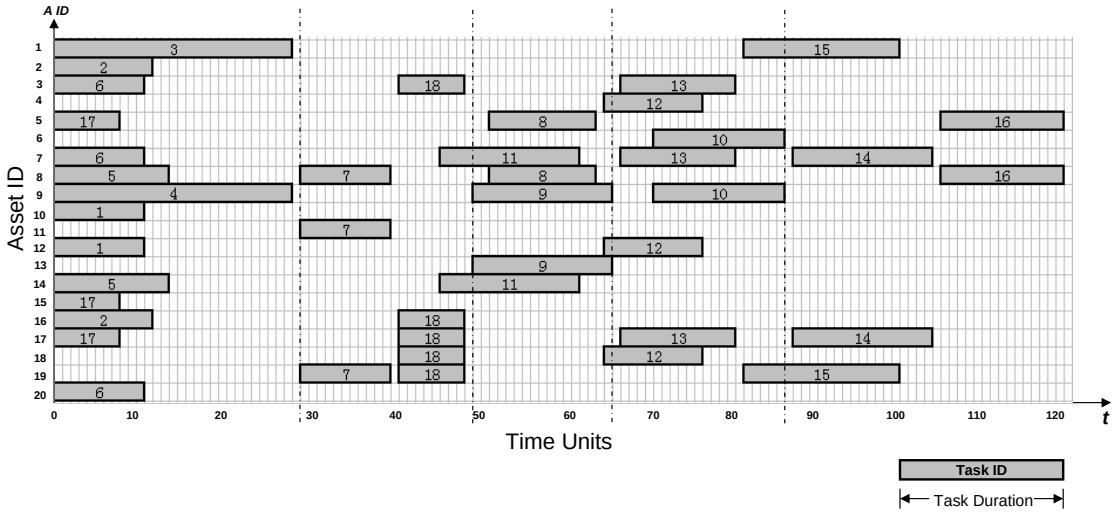


Fig. 17. The shifted schedule

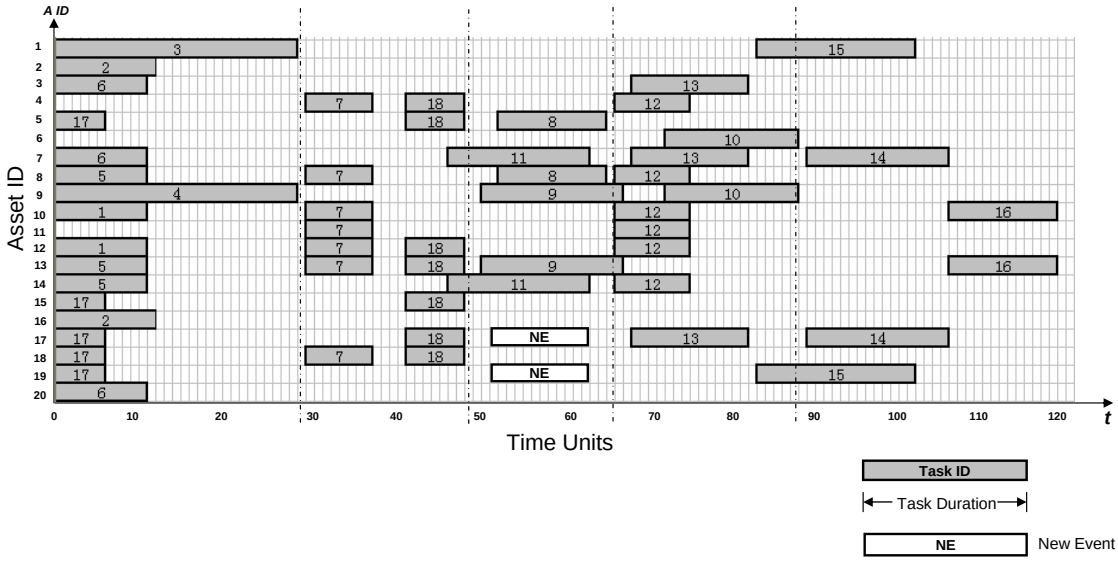


Fig. 18. The adjusted schedule

After illustrating the holonic scheduling process via a realistic mission example, it is concluded that the C^2 holonic reference architecture and the flexible scheduling scheme are an integration of centrality and autonomy; rigidity and flexibility; doctrine and adaptability, which enable a C^2 organization to possess the capability of handling mission changes, while keeping "unity of command" during mission execution.

However, we do realize that the flexibility of any scheduling scheme is limited by resource availability. As more redundant resources are available, a scheduling scheme can achieve more flexibility. We also note that, given a certain amount of resources, the higher the variety of schedules generated, higher is the degree of flexibility. However, the cost of adaptivity will be high potentially. A good scheduling scheme has to achieve a balance between performance and cost.

Future work needs to focus on (a) establishing

organizational synthesis models for hierarchy, heterarchy, and holarchy, for a (set of) mission environment(s); (b) exploring performance measures for different structures based on structural and mission environment models; and (c) application of the theory to existing and future systems (e.g., FORCENET, Expeditionary Strike Group).

ACKNOWLEDGMENT

This work was supported by the Office of Naval Research (ONR) under the contracts # N00014-00-1-0101 and # N00014-06-1-080.

REFERENCES

[1] "Expeditionary Warfare Staff Planning Brief presented at CWC Commander's Conference", *Tactical Training Group Pacific,* San Diego, CA.

[2] F. Yu, C. Meirina, S. Ruan, F. Tu, and K. Pattipati, "Integration of Holarchy and Holonic Scheduling Concepts for C^2 Organizational Design", *Proceedings of the 2005 CCRTS,* Washington DC, June, 2005.

[3] G.M. Levchuk, Y. N. Levchuk, C. Meirina, K. R. Pattipati, and D.L. Kleinman, "Normative Design of Organizations - Part III: Modeling Congruent, Robust, and Adaptive Organizations", *IEEE Trans. on SMC:Part A : Systems and Humans,* Vol. 34, No. 3, pp. 337-350, 2003.

[4] F. Yu, F. Tu, and K. R. Pattipati, "Congruent Organizational Design Methodology Using Group Technology and a Nested Genetic Algorithm", *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 36, no. 1, pp. 5-18, January, 2006.

[5] U. Aickelin and K. Dowsland, "An Indirect Genetic Algorithm for a Nurse Scheduling Problem," *Computers and Operational Research*, vol. 31, no. 5, pp 761-778, 2003.

[6] M. T. Jenson, "Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms", *IEEE Transactions on Evolutionary Computation*, vol. 7, no.5, pp. 503-515, Ocotber 2003.

[7] I. F. Sbalzarini, S. Muller, and P. Koumoutsakos, "Multiobjective optimization using evolutionary algorithms", *Center for Turbulence Research Proceedings of the Summer Program*, pp. 63-76, 2000.

[8] J. D. schaffer, "Multiple objective optimization with vector evaluated genetic algorithms", *In Grefenstette, J. J., editor, Proceedings of an International Conference on Genetic Algorithms and Their Applications,* pp. 93-100, 1985.

[9] P. Hajela and C. Y. Lin, "Genetic search strategies in multicriterion optimal design", *Structural Optimization,* vol. 4, pp. 99-107, 1992.

[10] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms", *Evolutionary Computation,* vol. 2, no. 3, pp. 221-248, 1994.

[11] J. Horn and N. Nafpliotis, "Multiobjective optimization using the niched Pareto genetic algorithm", *IlliGAL Technical Report 93005, Illinois Genetic Algorithms Laboratory,* University of Illinois, Urbana, Illinois, 1993.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective Genetic Algorithms NSGA II", *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-196, April, 2002.

[13] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: methods and applications", *Ph.D. thesis, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland,* Shaker Verlag, Aachen, Germany, ISBN 3-8265-6831-1, 1999.

[14] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy", *Evolutionary Computation,* vol. 8, pp. 149-172, 2000.

[15] C. A. Coello Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques", *Knowledge and Information systems*, vol. 1, no. 3, pp. 269-308, August 1999..

[16] C. A. Coello Coello, V. Veldhuizen and G. B. Lamont, *Evolutonary Algorithm for solving multi-objective problems,* Kluwer Acadamic publishers, New York, March 2002.

[17] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results", *Evolutionary Computation,* vol. 8, no. 2, pp. 173-195, 2000.

[18] S.-C. Lin, E. D. Goodman, and W. F. Punch, "A genetic algorithm approach to dynamic job shop scheduling problems",*Proceeding of International Conference on Genetic Algorithms, T. Back, Ed.,* pp. 481-488, 1997.

[19] C. Bierwirth and D. C. Mattfeld, "Production scheduling and rescheduling with genetic algorithms", *Evol. Comput.,* vol. 7, no. 1, pp. 1-18, 1999.

[20] H. G. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments", *Naval Res. Lab. Tech. Rep. AIC-90-001,* Washington, DC , 1990.

[21] F. Vavak, K. Jukes, and T. C. Fogarty, "Adaptive combustion balancing in multiple burner boiler using a genetic algorithm with variable range of local search", *Proc. Int. Conf. Genetic Algorithms, T. Back, Ed.,* pp. 719-726, 1997.

[22] C. L. Ramsey and J. J. Grefenstette, "Case-based initialization of genetic algorithms", *Proc. Int. Conf. Genetic Algorithms, S. Forrest, Ed.,* pp. 84-91, 1993.

[23] S. Yang, "Non-stationary problems optimization using the primal-dual genetic algorithm", *Proc. Congr. Evol. Comput.,* vol. 3, pp. 2246-2253, 2003.

[24] M. Farina, K. Deb and P. Amato, "Dynamic Multiobjective Optimization Problems: Test Cases, Approximations, and Applications", *IEEE Transactions on evolutionary computation,* vol. 8, no. 5, pp. 425-442, October 2004.

[25] P. Amato and M. Farina, "An ALife-Inspired Evolutionary Algorithm for Dynamic Multiobjective Optimization Problems", *8th on-line World Conference on Soft Computing in Industrial Applications WSC8,* http://wsc8.e-technik.uni-dortmund.de.CP, 2003.

[26] L. T. Bui, J. Branke, and H. Abbass, "Multi-objective optimization for dynamic environments", *The Artificial Life and Adaptive Robotics Laboratory ALAR Technical Report Series TR-ALAR-200504007,* Northcott Drive, Campbell, Canberra, Australia, 2005.

[27] K. Yamasaki, "Dynamic Pareto optimum GA against the changing environments", *Proc. Genetic Evolutionary Computation Conf. Workshop Program,* pp. 47-50, San Francisco, CA, July 2001.

[28] S. Y. Ho, L. S. Shu, and J. H. Chen, "Intelligent Evolutionary Algorithms for Large Parameter Optimization Problems", *IEEE Transactions on Evolutionary Computation,* vol. 8, no. 6, pp. 522-541, December 2004.

[29] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization", *Forrest, S., editor, Proceedings of the Fifth International Conference on Genetic Algorithms,* pp. 416-423, Morgan Kaufmann, San Mateo, California, 1993.

A Flexible Distributed Scheduling Scheme for Dynamic Expeditionary Strike Group (ESG) Mission Environments

Feili Yu
Sui Ruan
Candra Meirina
Prof. David L. Kleinman
Prof. Krishna R. Pattipati

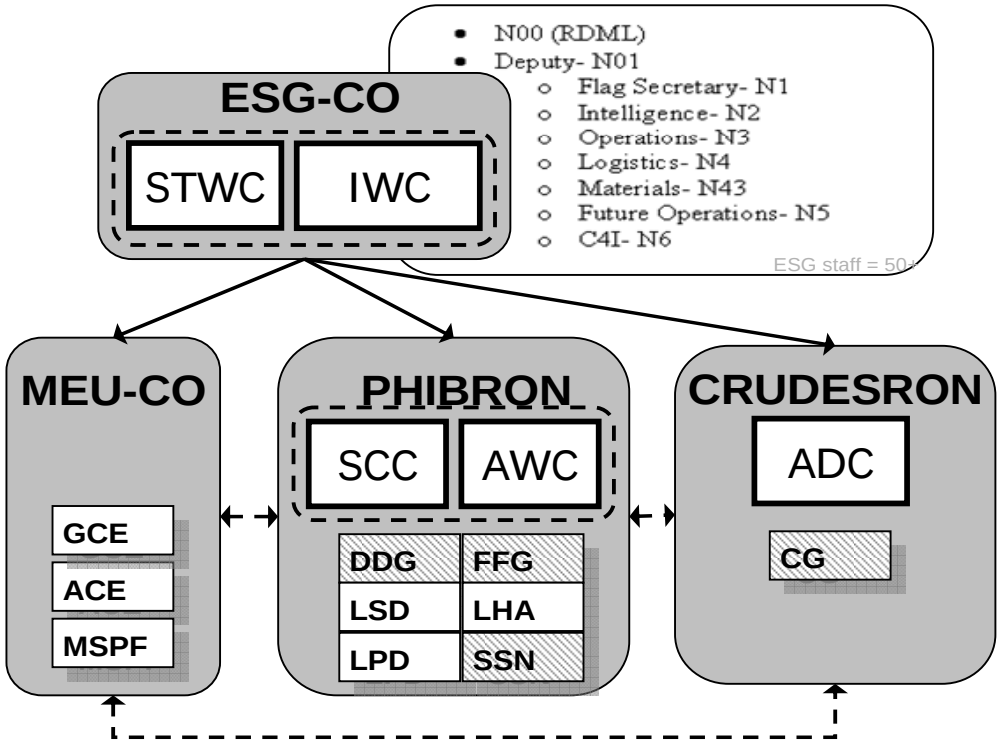
Dept. of Electrical and Computer Engineering
University of Connecticut
Contact: krishna@engr.uconn.edu (860) 486-2890

CCRTS 2006
June 20-22, 2006

Outline

- A Flexible Distributed Scheduling Scheme for Dynamic ESG Environments
  - ESG Planning Challenges
  - Multi-objective Model for Distributed Scheduling
  - Multi-Objective Optimization and Evolutionary Algorithms (MOEA)
  - Five-step Process for Flexible Distributed Schedule
  - Summary

Expeditionary Strike Group (ESG)



- ESG-CO:
  Expeditionary Strike Group Commander
- MEU-CO:
  Marine Expeditionary Unit Commander
- PHIBRON:
  Amphibious Squadron Commander
- CRUDESRON: AGEGIS Cruiser Commander

- ADC: Air Defense Commander
- AWC: Amphibious Warfare Commander
- IWC: Information Warfare Commander
- SCC: Sea Combat Commander
- STWC: Strike Warfare Commander

- ACE: Air Combatl Element
- GCE: Ground Combatl Element
- MSPF: Maritime Special Purpose Force
- DDG: Destroyer Ship
- FFG: Frigate Ship
- LSD: Dock Landing Ship
- LHA: Amphibious Assault Ship
- LPD: Amphibious Transport Dock Ship
- SSN: Submarine
- CG: Cruiser Ship

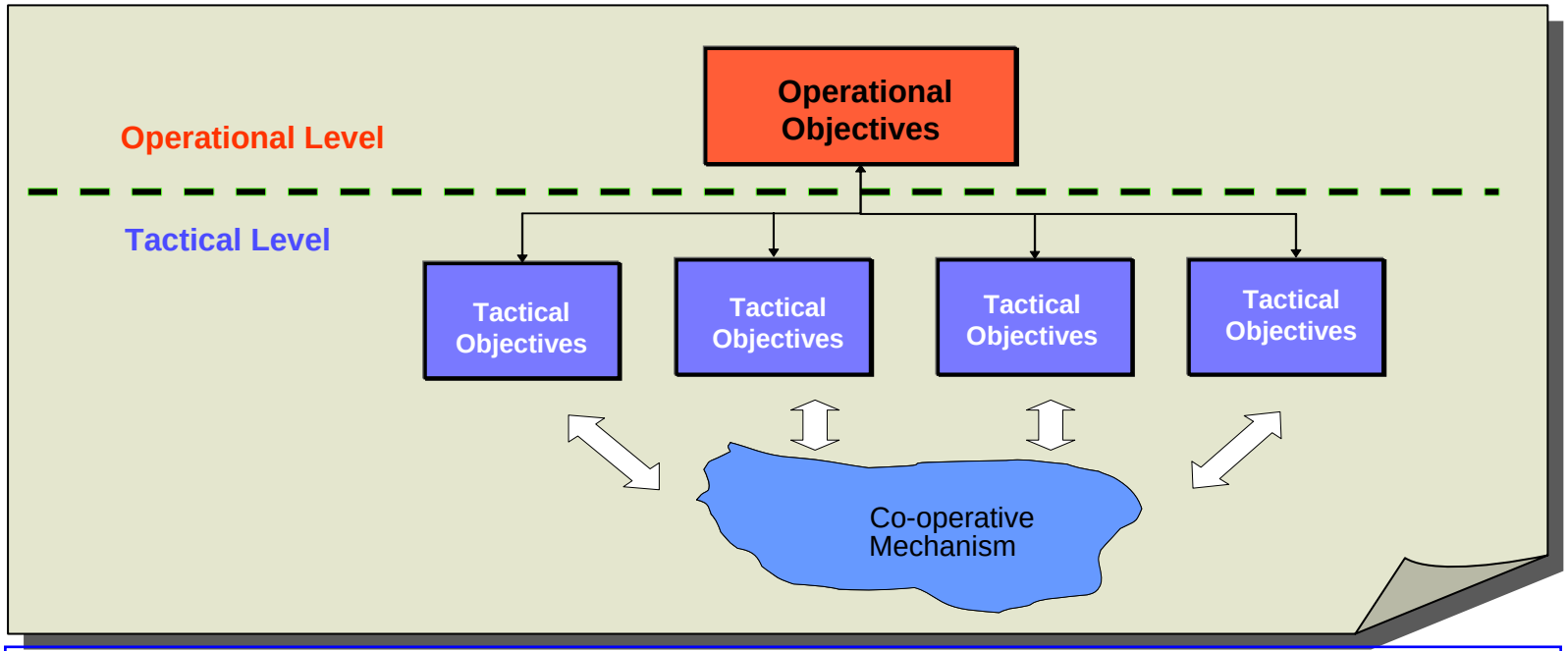
❑ An Expeditionary Strike Group (ESG) is a new US Navy task force that integrates navy warships and marines so that

- Assemble a specific package of assets to combat a specific threat
- leverage the synergistic effects of a unit that trained together for a broad set of missions
- build an organization that could either be as small task force or a larger joint task force

ESG Planning Challenges

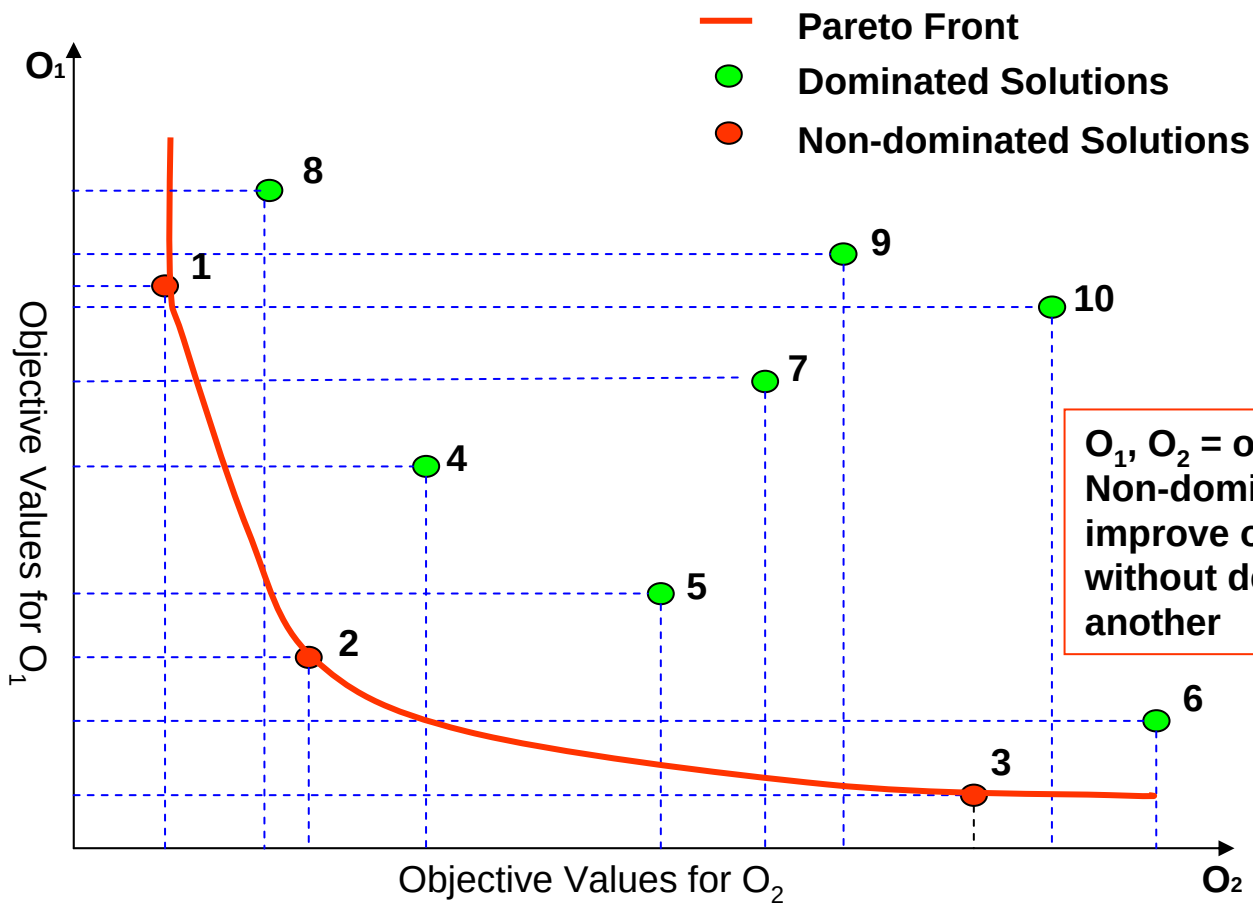
- **Scarcity of Resources**: ESG units are provided with limited resources to accomplish their objectives. The capabilities of resources may degrade after each operation (e.g., require maintenance) and some assets may breakdown due to incidents
- **Concurrent Operations**: conducting multiple concurrent operations that span the entire spectrum of missions is an integral part of an ESG. A complicating factor is that ESG units are being tasked with many unanticipated operations that are not planned for or scheduled a priori
- **Dispersed Operations**: ESG's assets are likely to be dispersed geographically to some location outside the immediate battle-space. At the same time, ESG is required to detach assets to support other commanders, and to assume control of additional assets from coalition partners
- **Conflicts between Local Priorities and Global Mission Objectives**: requires a new planning/scheduling scheme able to resolve priority conflicts among high-level and low-level ESG units

Multi-objective Model for Distributed Scheduling



- **Operational Level Objectives:**
  - Maximize Task Completion Accuracy ("mission success")
  - Minimize the Makes-pan of Mission Execution ("speed of command")
- **Tactical Level Objectives:**
  - Maximize Task Completion Accuracy
  - Allocate Resources to Minimize Task Execution Cost (Time including Coordination)
- **Co-operative Mechanism:**
  - Task announcement
  - Contracting
  - Coordinated Task Execution

Multi-objective Optimization (MO)



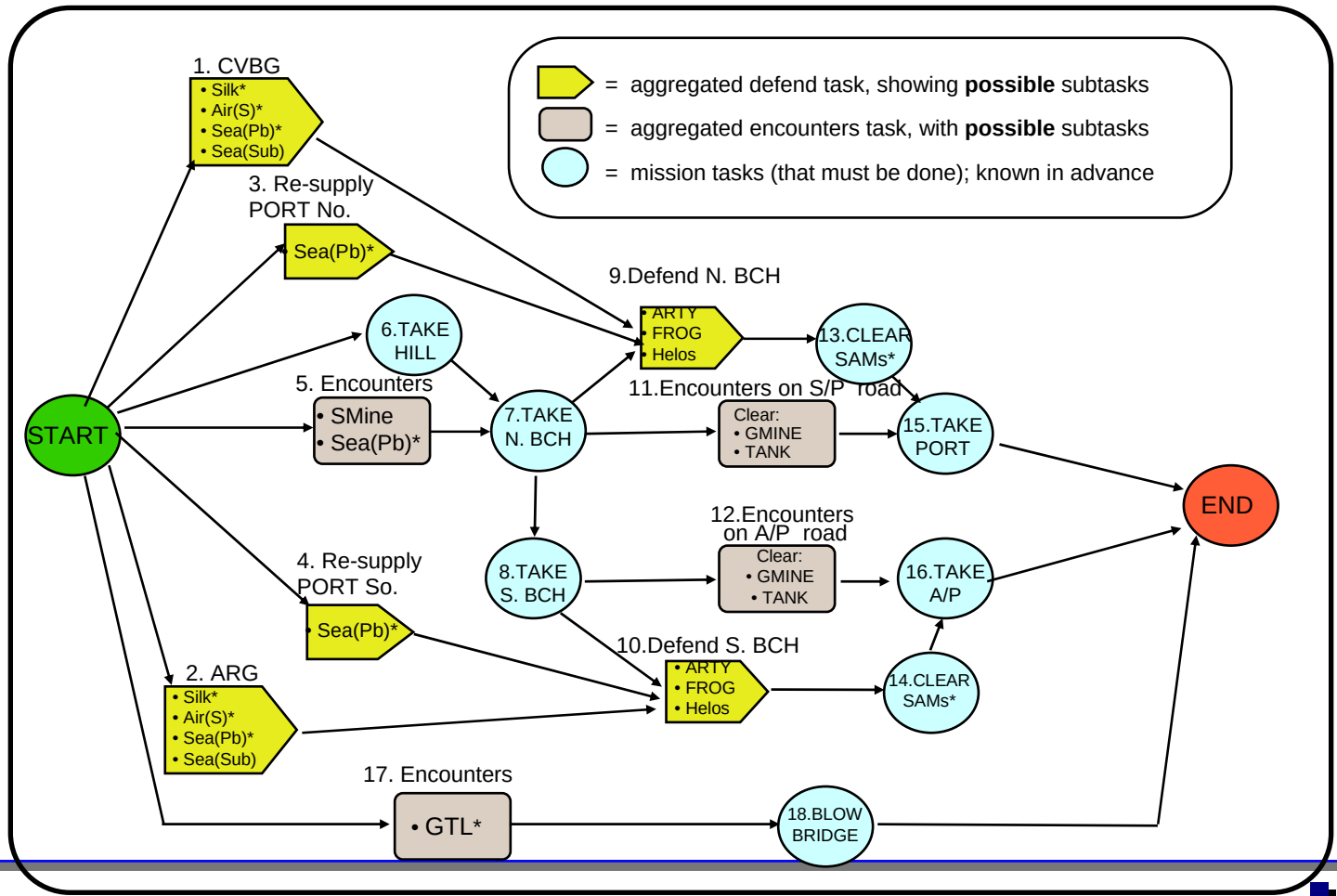
- Real world optimization problems often involve more than one objective, which may be conflicting with each other, thus, no global optimum can be found
- Pareto-based MO focuses on finding a set of promising solutions, namely, Pareto front, from which a solution can be chosen

Evolutionary Algorithms (EA) for MO

- Fitness Assignment
  - Pareto Ranking: an individual's rank corresponds to the number of individuals in current population by which it is dominated
  - Non-dominated Sorting: individuals are classified according to their non-dominance and fitness are shared in one class
- Elitism – *keep elitists in population*
  - Archiving: archive Pareto optimal solutions found so far during search
  - Recombination: include more high ranked individuals than low ranked in next generation
- Niching – *keep diversity of population*
  - Fitness Sharing: reduce the payoff of the fitness of individuals in one area
  - Crowding: only fraction of crowd individuals is selected to produce offspring

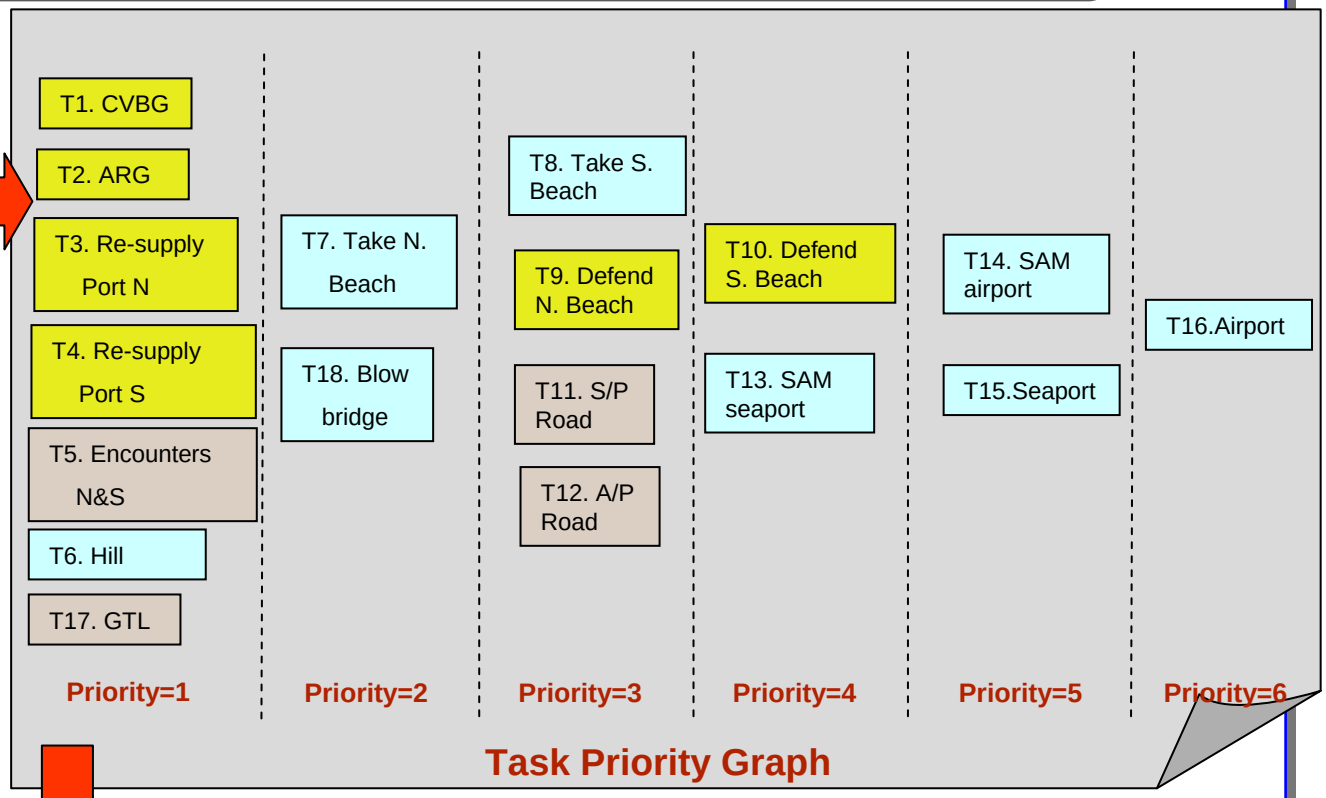
Illustrative Example

Mission: Assign a joint group of Navy and Marine Forces to capture a Seaport and take an Airport to allow for the introduction of follow-on forces. Utilize two suitable landing beaches: North Beach with a road leading to the Seaport; and South Beach with another road leading to the Airport. Intelligence sources report existence of hostile forces and potential counter-strikes.



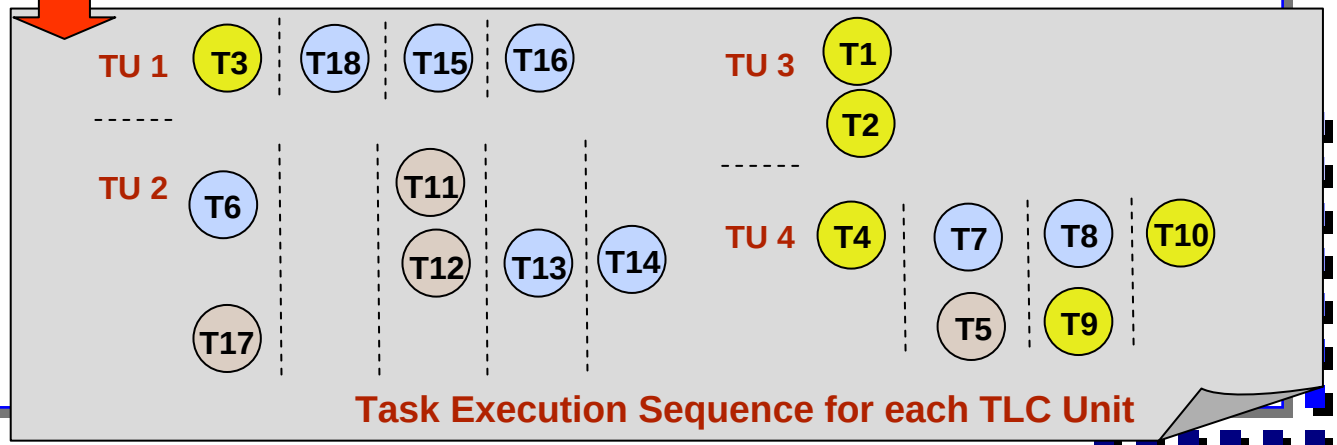
Solution Methodology for Flexible Distributed Scheduling - 1

Task precedence Graph



Step 1: Central Plan Decomposition

The task precedence graph is first decomposed based on task priorities, and then distributed among the tactical units



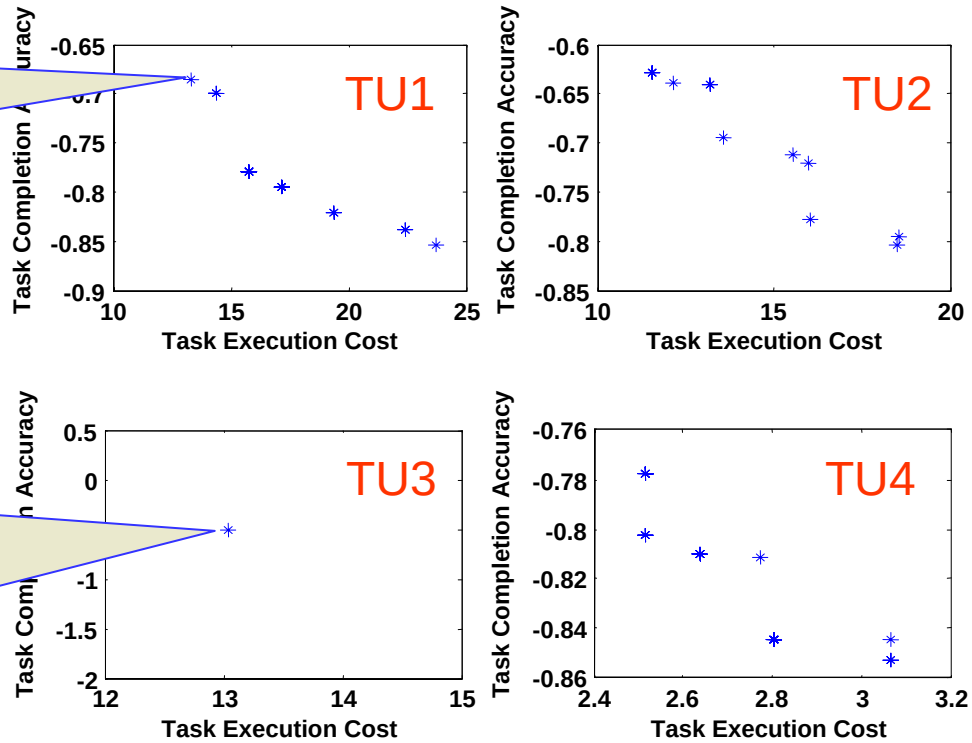
Solution Methodology for Flexible Distributed Scheduling - 2

Step 2: Tactical level Scheduling

- **Objectives:**
  - Maximize task execution accuracy
  - Allocate Resources to Minimize Task Execution Cost (Time including Coordination)

Corresponding to each solution in Pareto fronts, there is a local task-asset schedule

Not every TU can generate multiple local schedules. The more local schedules generated, the higher flexibility of global schedule will obtain

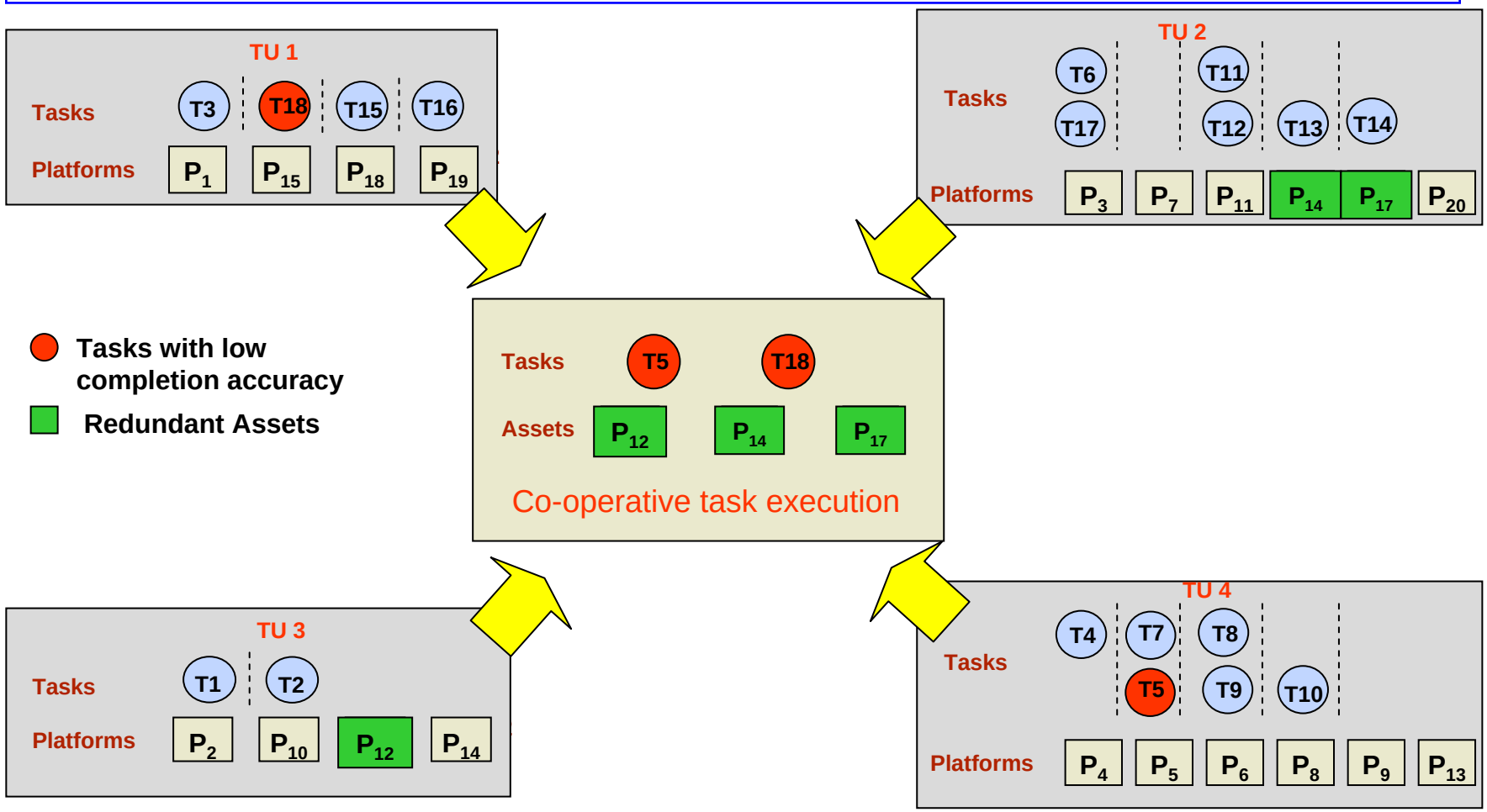


Pareto Fronts found for TU_1-TU_4

Solution Methodology for Flexible Distributed Scheduling - 3

Step 3: Co-operative Resource Re-deployment (single optimization problem)

- **Objectives:**
  - Re-deploy resources to coordinate on tasks with low accuracy
  - Maximize the mean task completion accuracy



Solution Methodology for Flexible Distributed Scheduling - 4

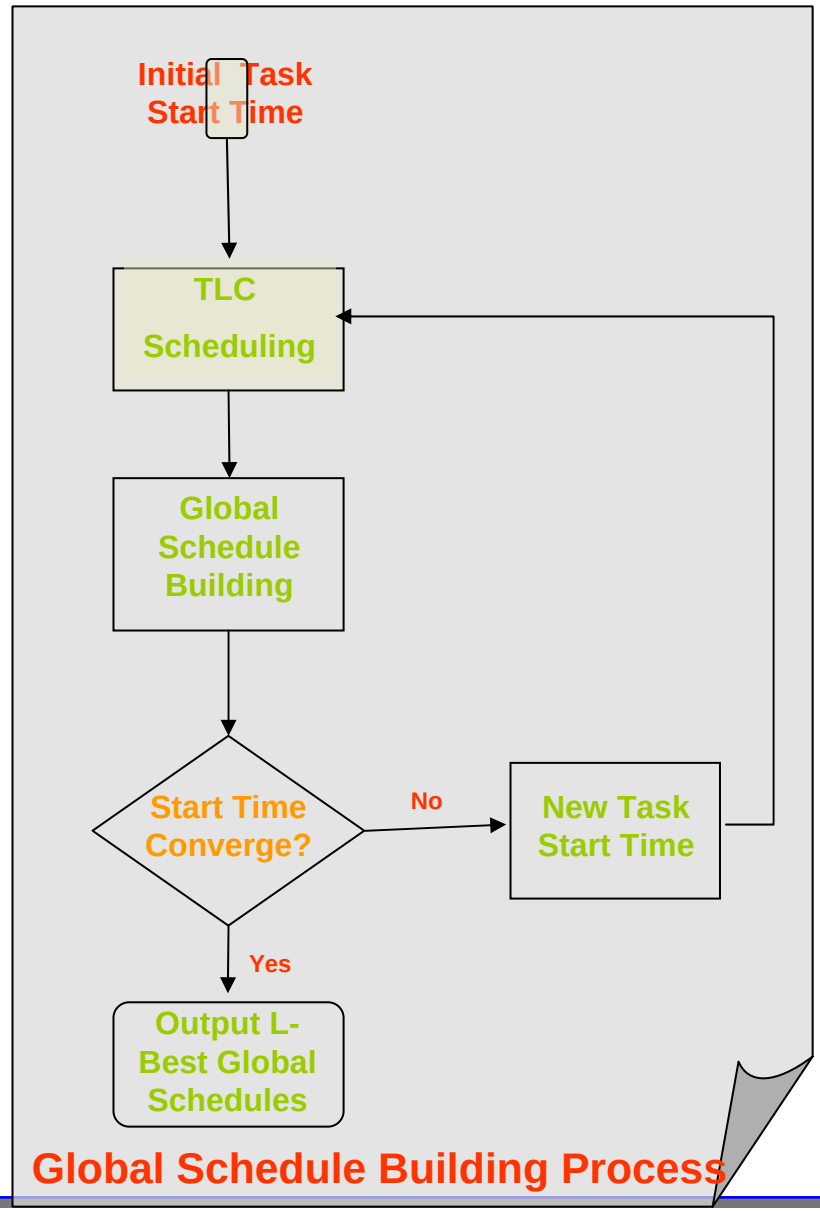
Step 4: Operational Level Global Schedule Building

- **Objectives:**
  - optimize operational level objectives (mission success and speed of command)
  - assemble local schedules into a global schedule
  - resolve conflicts among local schedules and ensure the feasibility of global schedule
- **Global Schedule Building Process:**
  - *Right-shifting*

    local TU schedules are reassembled according to the task graph. The start times of tasks that have precedence constraints will be shifted to a time when all its preceding tasks have been accomplished
  - *How to achieve the stability of schedule?*

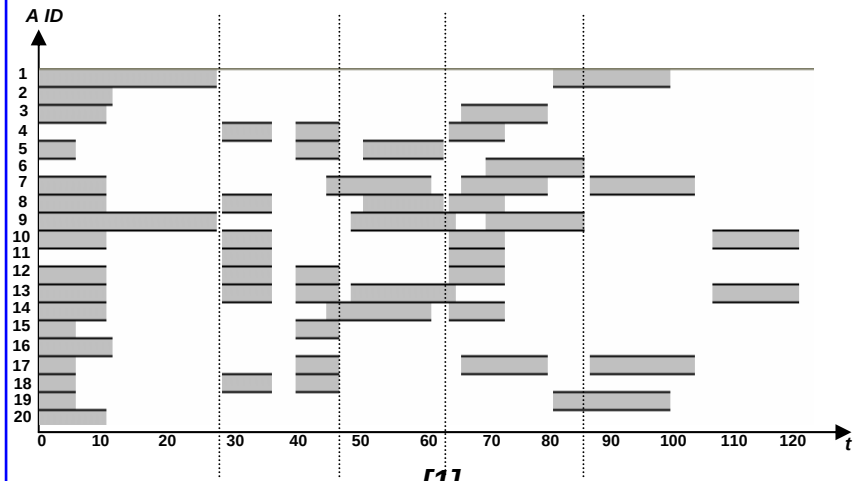
    Iteratively right-shift using new start times at each iteration till the schedule converges.
- **Outputs:**

  L-neighboring schedules

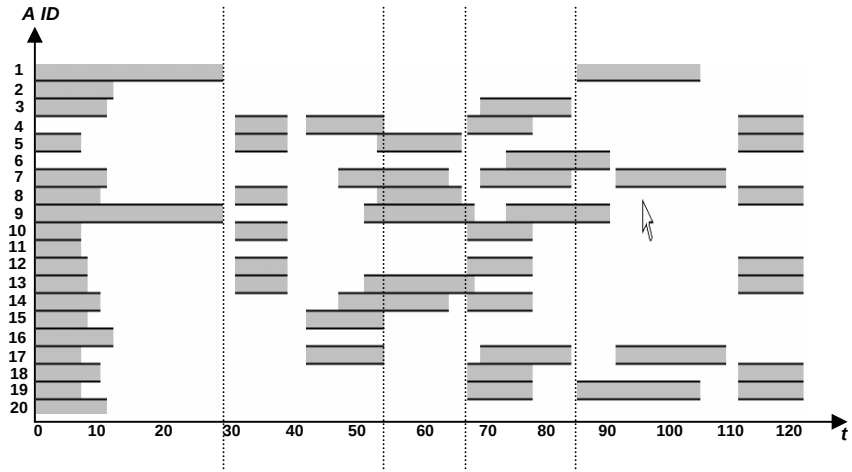


Global Schedule Building Process

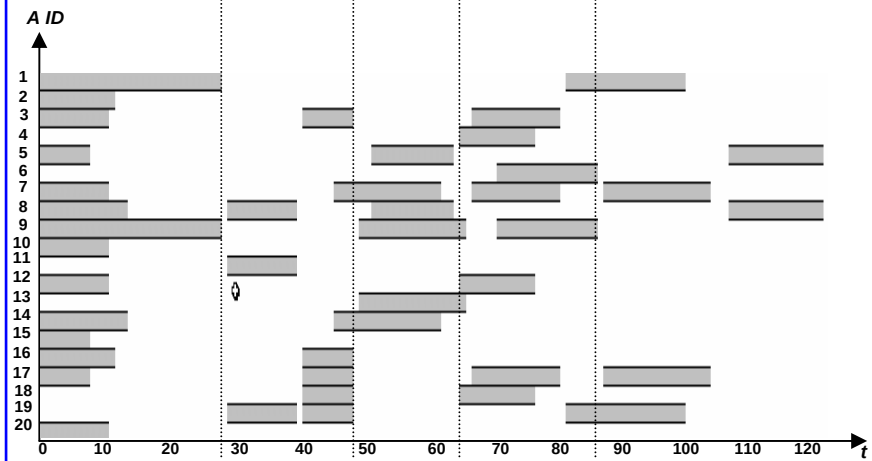
L-Neighboring Schedules



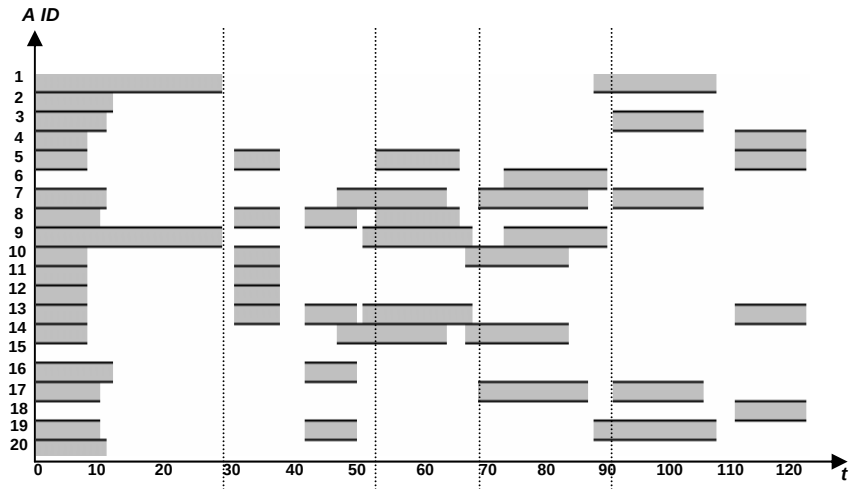
[1]



[2]



[3]

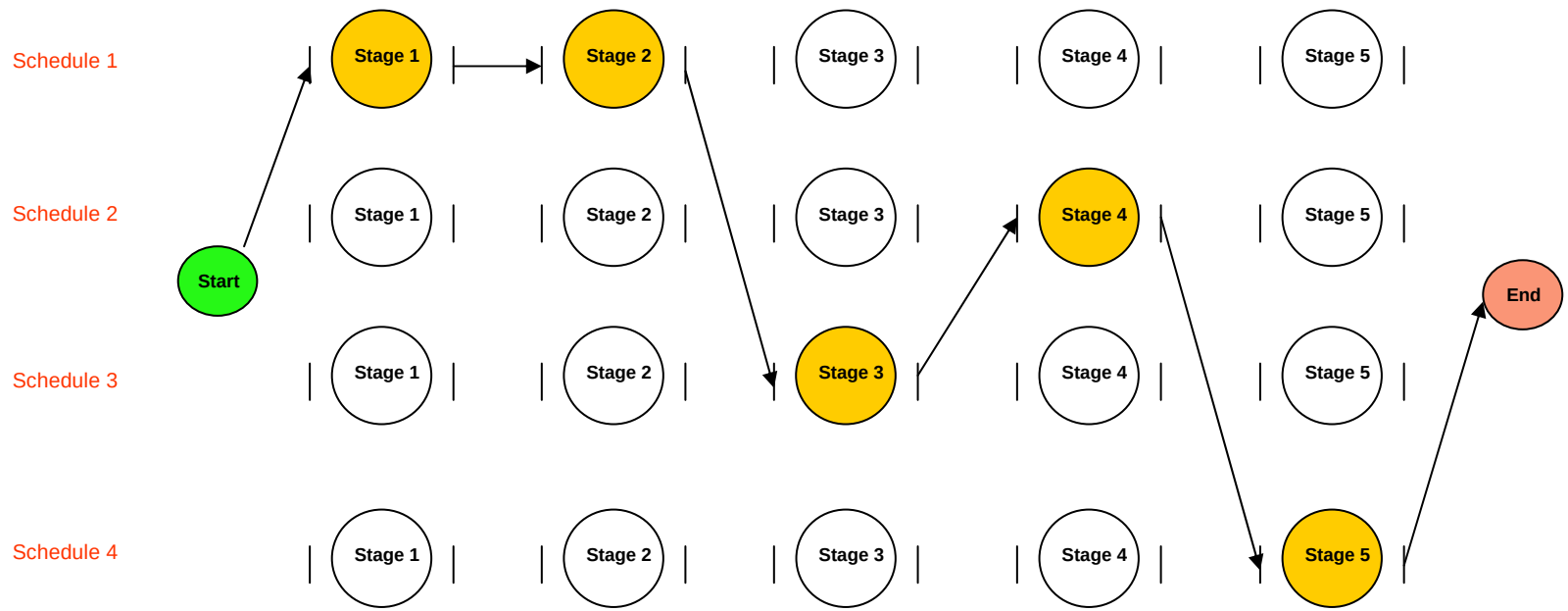


[4]

Solution Methodology for Flexible Distributed Scheduling - 5

Step 5: Adaptive Scheduling

- **Objectives:**
  - adapt to mission changes (new tasks, asset breakdowns, unit unavailability,…)
- **The global schedule can interact with the environment in the following way**
  1. **divide L-Neighboring schedules into several stages**
  2. **monitor mission execution and check feasibility of the current schedule**
  3. **Once infeasibility is detected, search among alternative neighboring schedules at the same stage until a feasible schedule is found**
  4. **instructs lower level tactical units to adapt to schedule changes**



Summary

- A flexible distributed scheduling scheme for dynamic ESG environments
- Multi-Objective Evolutionary Algorithm (MOEA) is employed to generate L-Neighboring schedules
- Final schedule is obtained by assembling different stages of L-Neighboring schedules in order to adapt to mission changes
- Insights into flexible scheduling
  1. Flexibility is limited by resource availability
  2. Higher the variety of generated schedules, higher the flexibility

Questions?