SEKE 2014

Practical Human Resource Allocation in Software Projects Using Genetic Algorithm

Jihun Park, Dongwon Seo, Gwangui Hong,
Donghwan Shin, Jimin Hwa, Doo-Hwan Bae

KAIST, South Korea

Outline

- Introduction
- Problem Definition
- Practical Considerations
- Genetic Algorithm for Human Resource Allocation
- Case Study
- Related Work
- Conclusion
- Discussion

Introduction

- Software planning is becoming more complicated as the size of software project grows.

- Software project managers can significantly benefit from the human resource allocation technique.

- Existing human resource allocation approaches only focused on minimizing the project cost.

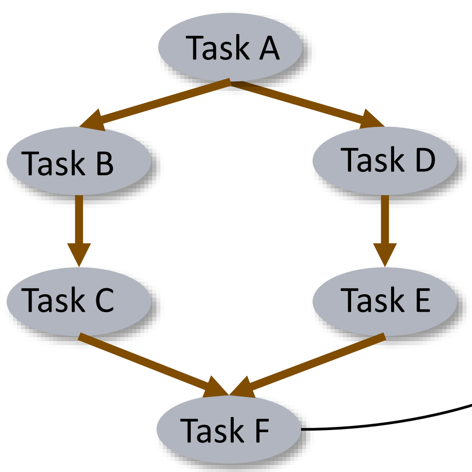
Research Goal

- We elicit the practical considerations on human resource allocation problem with a group of software project experts.

- We then suggest a novel Genetic Algorithm (GA) satisfying the practical considerations.

Problem Definition

- Tasks
  - Defined by type, effort, and precedence relationship
- Developers
  - Defined by staff level and ability level.

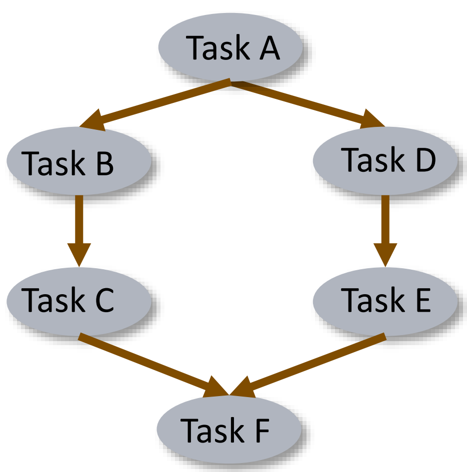


Task information

Task Name	Task F
Task Type	Test
Effort needed	200 MH
Preceding Tasks	Task C, Task E

Problem Definition

- Tasks
  - Defined by type, effort, and precedency relationship
- Developers
  - Defined by staff level and ability level.



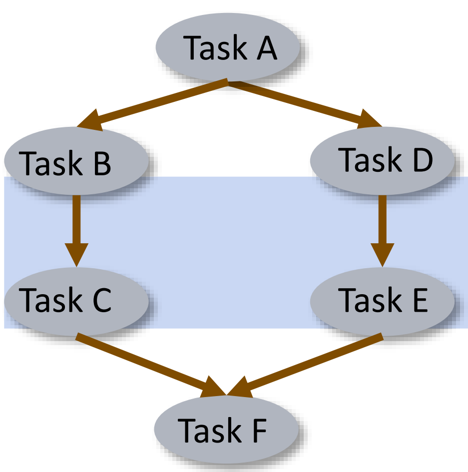
Task information

Name	Staff level	Analy sis	…	Test
 Tom	Engin eer	0.7		1.0
 Jane	Mana ger	1.0		1.2
 April	Direct or	1.5		0.8

Developer information

Problem Definition

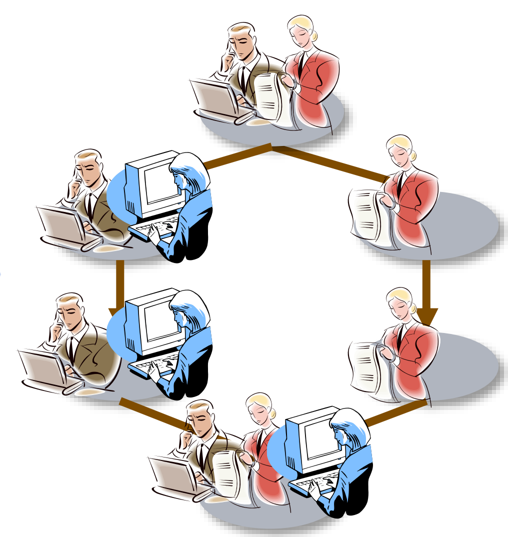
- Tasks
  - Defined by type, effort, and precedency relationship
- Developers
  - Defined by staff level and ability level.



Task information

Name	Staff level	Analysis	…	Test
Tom	Engineer	0.7		1.0
Jane	Manager	1.0		1.2
April	Director	1.5		0.8

Developer information



Developer Assignment

Practical considerations

- Eliciting practical considerations with a group of software experts from..
  - Military research and development company
  - Software process consulting company
  - University

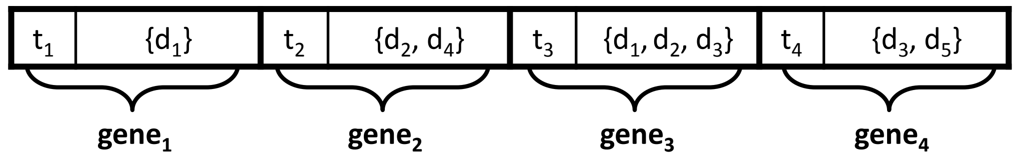
C1. Short project plan	The basic objective of human resource allocation. The plan should be finished within minimum timespan.
C2. Minimization of multitasking time	If a developer work for multiple tasks at the same time, productivity will decrease.
C3. Assignment on relevant tasks	Assigning a developer to both of pre-task and post-task is efficient in terms of minimizing the context-switching cost.
C4. Balance of allocation	Task size and staff level should be considered in the allocation.

Genetic Algorithm

- Evolutionary search-based algorithm

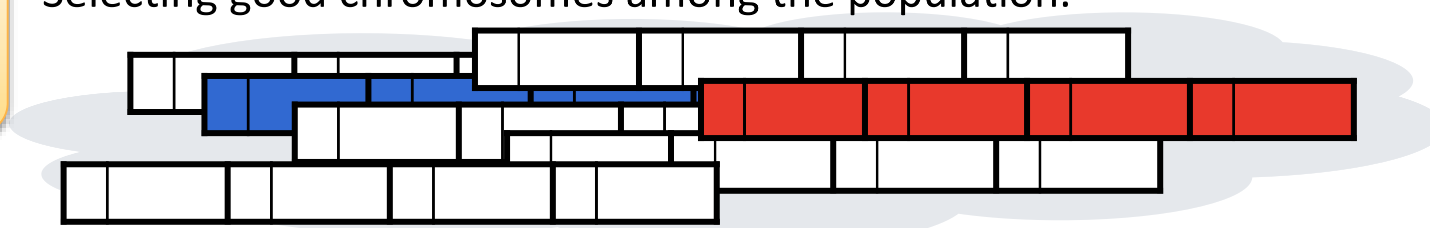
Representation

Representing an assignment result as a chromosome.



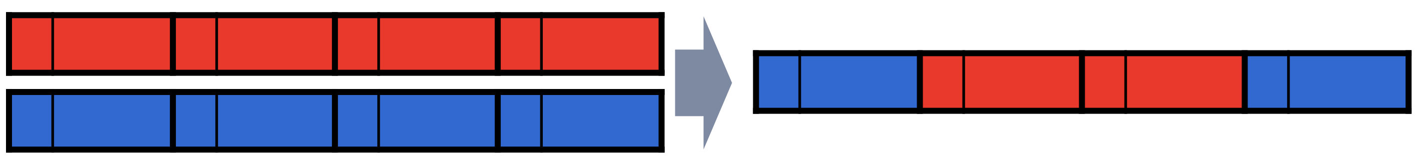
Selection

Selecting good chromosomes among the population.



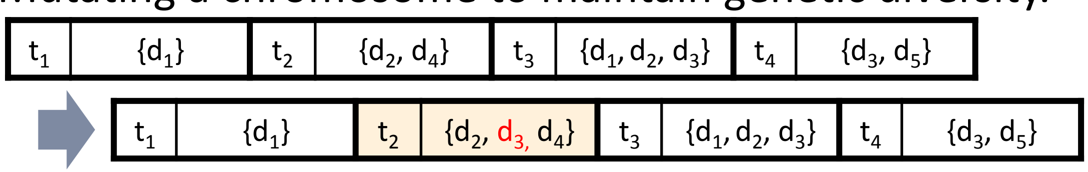
Crossover

Making a new chromosome from two parent chromosomes.



Mutation

Mutating a chromosome to maintain genetic diversity.



How chromosomes evolve?

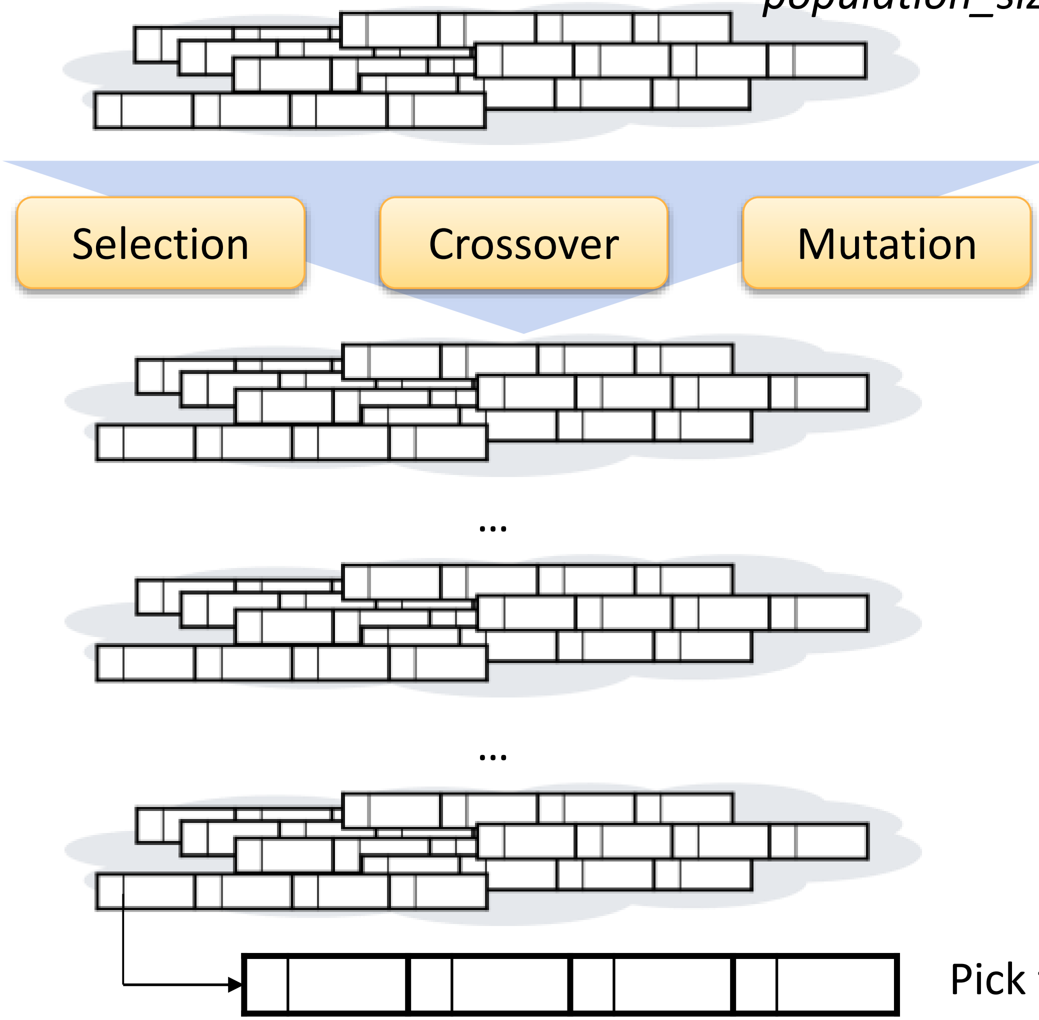
A population has
population_size chromosomes

Initial population
(randomly generated)

2nd generation

nth generation

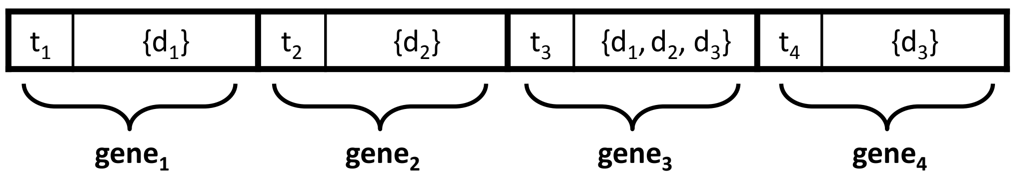
*Generation_count*th
generation



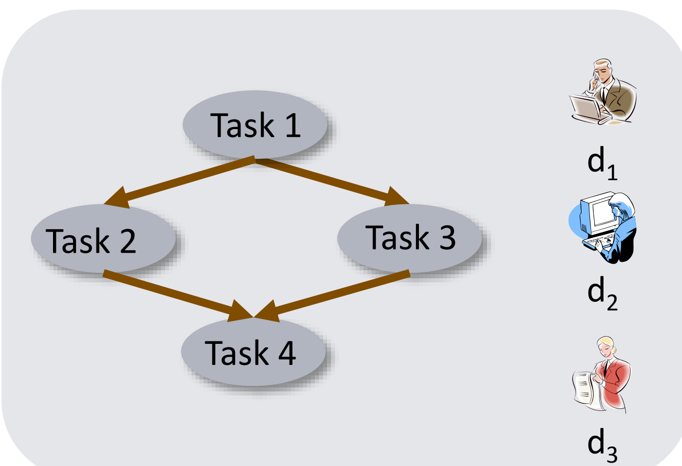
Pick the best one.

Representation

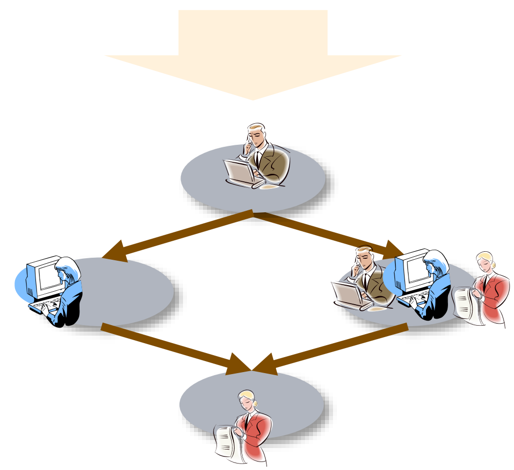
- Each gene contains a task and a set of developers.
  - Gene n represents a set of developers assigned to the task n



An example of chromosome representation



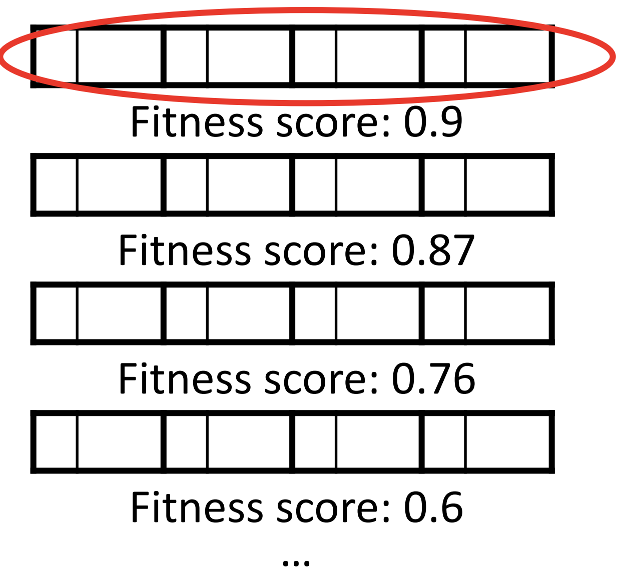
Task and developer information



Assignment example of the chromosome

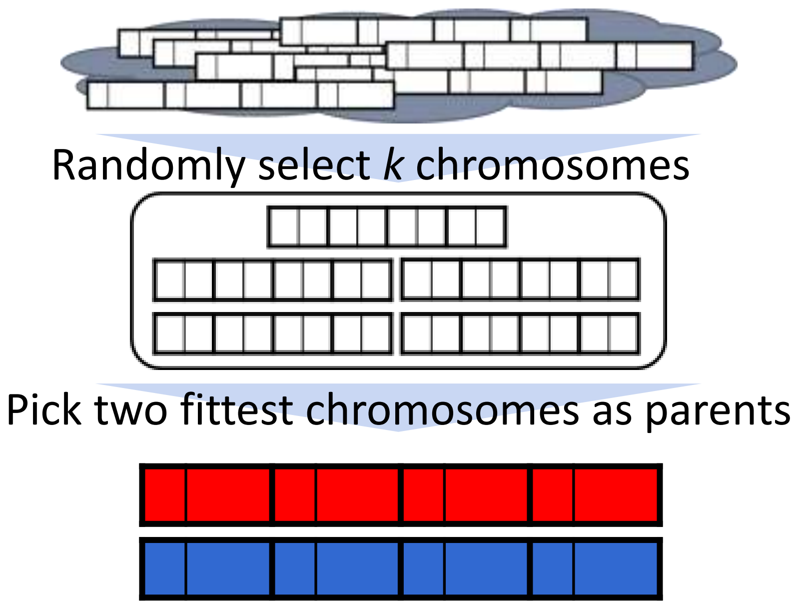
Selection

- Each chromosome is evaluated by ***fitness function***
- Elitism selection
  - Keep the best chromosome until the next generation.
- Tournament selection
  - Select two parent chromosome for a new chromosome.



Elitism selection

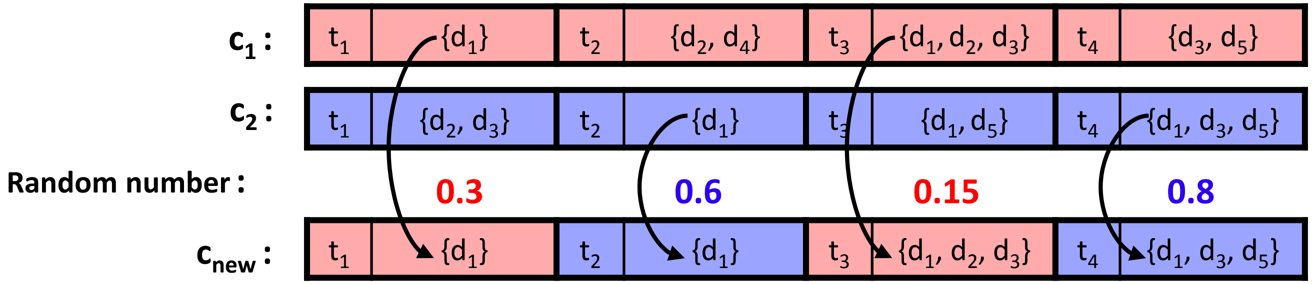
Keep the fittest chromosome to the next generation.



Tournament selection

Crossover

- Uniform crossover
  - Using parent chromosomes which are selected by tournament selection process.
  - Generating a new chromosome for the next generation.
  - Randomly taking a gene among two genes from the parents.



Mutation

- With a certain probability of the mutation (*mutation rate*), each gene is mutated.
- Three mutation operators
  - Assigning a random developer to the task.

Before mutation :	t_1	$\{d_1\}$	t_2	$\{d_2, d_4\}$	t_3	$\{d_1, d_2, d_3\}$	t_4	$\{d_3, d_5\}$
After mutation :	t_1	$\{d_1\}$	t_2	$\{d_2, d_3, d_4\}$	t_3	$\{d_1, d_2, d_3\}$	t_4	$\{d_3, d_5\}$

  - Removing a random developer from the task.

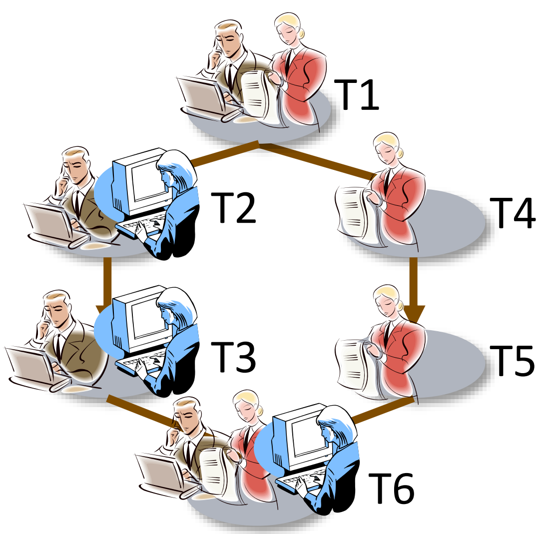
Before mutation :	t_1	$\{d_1\}$	t_2	$\{d_2, d_4\}$	t_3	$\{d_1, d_2, d_3\}$	t_4	$\{d_3, d_5\}$
After mutation :	t_1	$\{d_1\}$	t_2	$\{d_2\}$	t_3	$\{d_1, d_2, d_3\}$	t_4	$\{d_3, d_5\}$

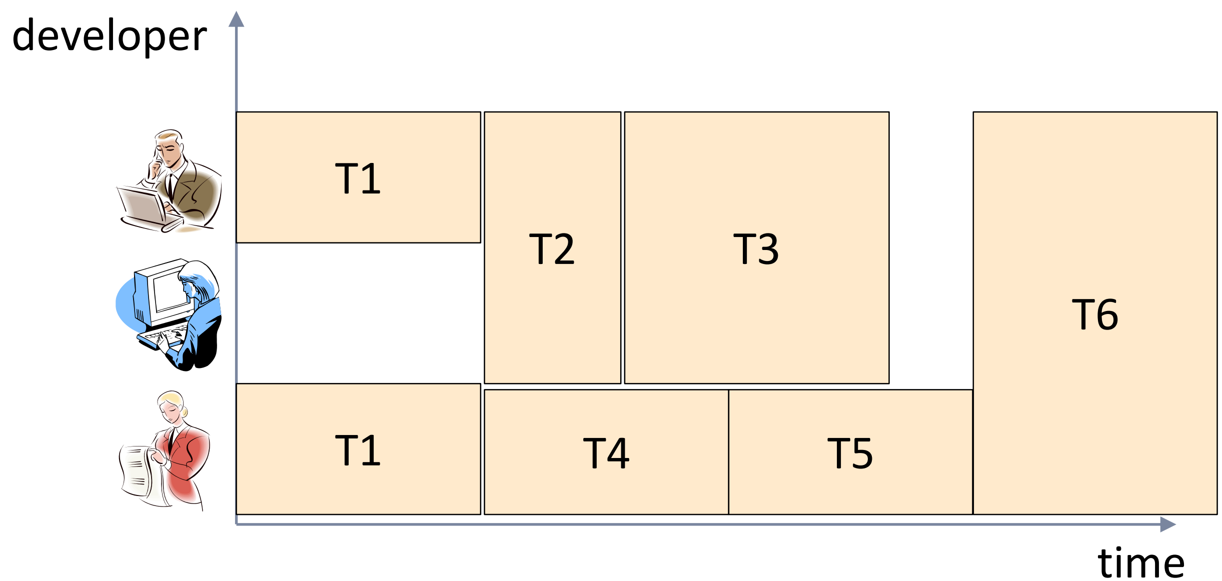
  - Replacing an assigned developers with a random developer.

Before mutation :	t_1	$\{d_1\}$	t_2	$\{d_2, d_4\}$	t_3	$\{d_1, d_2, d_3\}$	t_4	$\{d_3, d_5\}$
After mutation :	t_1	$\{d_1\}$	t_2	$\{d_2, d_3\}$	t_3	$\{d_1, d_2, d_3\}$	t_4	$\{d_3, d_5\}$

Scheduling simulation

- The fitness function is calculated based on the scheduling simulation.
- Principles
  - At every time tick, assigned developers reduce the remaining MH of a task.
  - Developers start to work for a task, if every pre-task of the task is finished.





Fitness Function

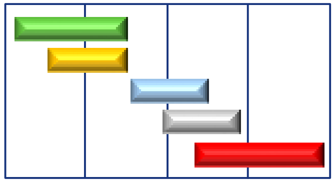
- The fitness function evaluates a chromosome by calculating *fitness score*.

- The fitness function reflects practical considerations.

- The fitness score is weighted sum of four sub-scores.
  - Cost Minimization (CM) score
  - Concentration Efficiency (CE) score
  - Continuity Consideration (CC) score
  - Balance of Allocation (BA) score

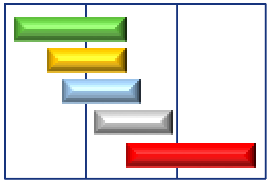
Fitness Function (cont'd)

- Cost Minimization (CM) score
  - Assessing whether the solution finishes early.
  - Comparing the timespan of the given solution with an ideal timespan.



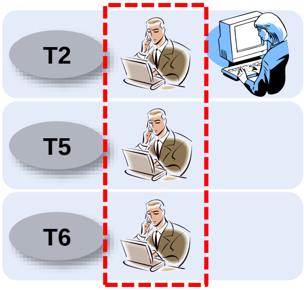
Taking 100 days

<

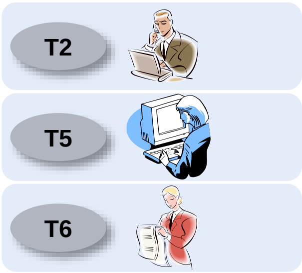


Taking 80 days

- Concentration Efficiency (CE) score
  - Assessing the burden of multitasking.

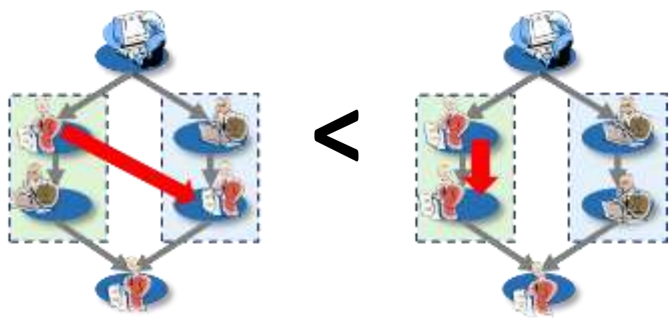


<

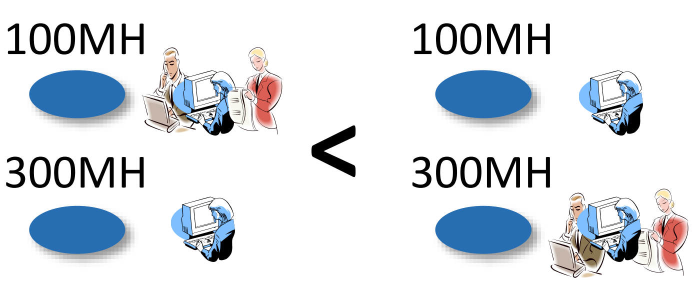


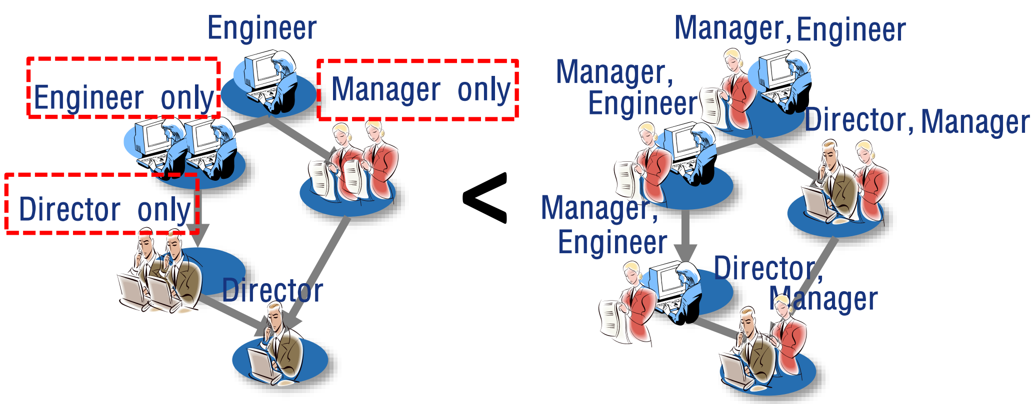
Fitness Function (cont'd)

- Continuity Consideration (CC) Score
  - Assessing consideration on the precedence relationships between tasks.



- Balance of Allocation (BA) Score
  - Assessing how evenly the developers are allocated.





Case Study

- Assessing how well our GA reflects the practical considerations.

- Comparing the results when GA only considers cost minimization($Case_{time}$) and when considering the all objectives ($Case_{all}$).

- Weights for fitness score $\{w_{CM}, w_{CE}, w_{CC}, w_{BA}\}$:

$Case_{time}$: $\{1, 0, 0, 0\}$, $Case_{all} = \{0.25, 0.25, 0.25, 0.25\}$

Case Study

- Experimental setup
  - Three experiment sets
    - Set1: 11 tasks / 7 developers
    - Set2: 11 tasks / 10 developers
    - Set3: 21 tasks / 10 developers
  - Population size: 100, Generation count: 400,
  - Mutation rate: 0.05

ID	$type_i$	$effort_i$	PT_i
t_1	Analysis	400	
t_2	Design	320	1
t_3	Design	240	1
t_4	Design	240	1
t_5	Implementation	240	2
t_6	Implementation	600	3
t_7	Implementation	160	4
t_8	Test	100	5
t_9	Test	80	6
t_{10}	Test	70	7
t_{11}	Test	80	8,9,10

Task set t_1

ID	sl_j	$ability_j^k$			
		analysis	*design*	*implementation*	*test*
d_1	3	1.25	1	1.25	1.25
d_2	3	1.25	1	1.25	0.75
d_3	2	0.75	0.75	1	1
d_4	2	0.75	1	1	0.75
d_5	1	1	0.75	0.75	1
d_6	1	0.75	1	0.75	0.75
d_7	1	1	0.75	1	0.75

Developer set d_1

Case Study

Metric	Set1 #task=11 #dev=7		Set2 #task=11 #dev=10		Set3 #task=21 #dev=10	
	All	Time	All	Time	All	Time
Time (h)	342.71	322.17	239.70	225.04	846.30	744.14
Multitasking Time (h)	28.96	101.76	12.49	54.94	58.38	404.93
# no precedence assignments	7.91	18.31	10.92	27.08	15.11	41.17
# tasks only one level asisgned	3.73	3.05	1.69	3.07	5.74	6.56
Mean (# assigned devs / $effort_i$)	2.15e-02	3.50e-02	2.80e-02	4.70e-02	1.07e-02	1.88e-02
Variance (# assigned devs / $effort_i$)	1.71e-04	7.47e-04	2.69e-04	1.48e-03	4.29e-05	1.24e-04

Experiment results

- Time: $Case_{time} < Case_{all}$
- Multitasking time: $Case_{time} > Case_{all}$
- # no precedence assignments: $Case_{time} > Case_{all}$
- # tasks only one level assigned: $Case_{time} > Case_{all}$
- Evenness of allocation: $Case_{time} < Case_{all}$

Overall, our GA reflect practical considerations better than an approach only considering cost minimization.

Related Work

- Chang et al. (IST 2008) and Chen et al. (TSE 2013)
  - GA/ACO (Ant colony optimization) techniques considering three-dimensional array with time, tasks, employee axes.
  - They only concentrated on minimizing cost in terms of time and money.

- Kang et al. (SPE 2011)
  - A constraint-based approach considering constraints affecting project schedule.
  - They assumed each program modules can always be developed in parallel.

Conclusion

- We *elicit practical considerations* for human resource allocation problem with a group of experts.

- We *design a genetic algorithm* reflecting the practical considerations by *encoding them in fitness function*.

- Our GA generates a practical human resource allocation *considering multitasking time, precedence relationship, and balance of allocation*.

Discussion

- Threats to validity
  - Many previous approaches used skill sets to represent required capability of a task, but we use task types.
  - Our approach generates only one fittest solution. MOEA approach can be used to generate more than one solution.

- Future work
  - Finding optimal parameters for GA.
  - Identifying more practical issues.
  - Studying the applicability of our approach in real-world.

Thank you for listening

Practical Human Resource Allocation in Software Projects Using Genetic Algorithm

Jihun Park, Dongwon Seo, Gwangui Hong,
Donghwan Shin, Jimin Hwa, Doo-Hwan Bae

KAIST, South Korea