Computational Intelligence II Genetic Algorithms

Master of Technology in Intelligent Systems

GA Optimizer for Project Release Scheduling and Resource Allocation

Sep-Oct 2014

Team Members

GU Zhan (Sam)	A0107682A
LIU Jie (Judy)	A0107579U
Francisco Liwa	A0107538A

Abstract

We developed an IT project release schedule optimizer using Genetic Algorithm, considering various real world constraints, including human resources, lead time and software version compatibility. The goal for optimizer is to shorten the overall projects' life span as well as individual project's life spans for a group of concurrently running projects. The implementation tools used are Microsoft Excel and Evolver.

Keywords: Genetic Algorithm, Project Release Management, Optimization, Evolver

Table of Content

1. Problem Introduction

Release Management is the process of managing software (IT project) releases from development stage to software release in production. Software products (with different versions/upgrades) are typically in an ongoing cycle of planning, development, various testing, and production release. [1]

There is growing complexity of dynamic project dependencies, maximum available resources, and large number of parallel running projects, together with various moving/changing influencing factors to take into consideration. These factors must fit together seamlessly to guarantee success and long-term value of multiple running projects as a whole.

We developed a simplified GA optimizer to conduct multiple project releases scheduling (to shorten make-span of both individual project and all projects as a whole), considering several resource constraints, including available human resources, testing environments, and software version conflicts among all projects.

This pilot model will serve as a base to tackle further sophisticated release management requirements in real Information Technology industries discussed in ***Section 4.2 Future Work***. With ISS guidance, we aim to fully develop the enhanced intelligent system, and to submit a paper to the 19th ***KES International Conference on Knowledge-Based and Intelligent Information & Engineering Systems***, held in Sept 2015 in Singapore.

2. Modeling - Using Evolver / Excel Tool

2.1. Background Knowledge and User Configurable Inputs

For a given project, below data/formulas are known/predefined:

- The required 'Effort (in Person-Days)' of different types of human resources (Knowledge Engineer/Software Engineer) for different projects phases (Plan/Dev/SIT/UAT/NFT/Prod)
- The total available number of different types of human resources (FTE Full Time Employee) (Knowledge Engineer/Software Engineer) (In this model, we defined max 8 Knowledge Engineers, and 24 Software Engineers)
- The number of projects in scope (In this model, we defined 15 projects)
- The actual 'Project Phase Span (in Calendar-Days)' in a given project phase is calculated based on formula: MAX(KE Effort/Allocated KE FTE, SE Effort/Allocated SE FTE), with assumption that work stream of Knowledge Engineer and Software Engineer can work in parallel in a certain project phase.
- The Code Base violation (software version conflict) can happen in project phases: SIT/UAT/NFT/Prod; But won't happen in Plan/Development phases.

2.2. Chromosome Representation

GA permutation happens for

- KE FTE and SE FTE resource allocations for different projects
- The Lead / Idle time prior to different project phases (Plan/UAT/NFT/Prod)
  Because the Code Base violation won't happen in Plan and Development phases, which means Development and SIT phases can start immediately after the completion of their respective predecessor phase. Thus in this model setting, the 'Lead' prior to the Development and SIT phases are fixed to zeros, also reducing the GA search space as well.

Chromosome	Resource Allocation / Full Time Employee (FTEs)		Lead (Calendar-Day)			
Project Name	Knowledge Engineer	Software Engineer	Lead prior to Plan	Lead prior to UAT	Lead prior to NFT	Lead prior to Prod
Project-A01	0.50	1.50	0	0	1	0
Project-A02	0.50	1.50	12	1	0	1
Project-A03	0.50	1.50	23	0	1	0
Project-A04	0.50	1.50	34	1	0	1
Project-A05	0.50	1.50	45	0	1	0
Project-A06	0.50	1.50	56	1	0	1
Project-A07	0.50	1.50	67	0	1	0
Project-A08	0.50	1.50	78	1	0	1
Project-A09	0.50	1.50	12	0	1	0
Project-A10	0.50	1.50	23	1	0	1
Project-A11	0.50	1.50	34	0	1	0
Project-A12	0.50	1.50	45	1	0	1
Project-A13	0.50	2.00	56	0	1	0
Project-A14	0.50	2.00	67	1	0	1
Project-A15	1.00	2.00	78	0	1	0
Average	0.53	1.60				
Total	8	24	630	7	8	7
	<=	<=				
Available FTE	8	24	<--- Hard constraint			

2.3. Fitness Function Design

We consider below objectives to design integrated fitness function:

- To shorten entire projects' life span (From earliest project's Start-Date till latest project's End-Date)
- To shorten (average) individual project's life span
- To schedule important projects to start earlier (Importance is based on project cost.)
- To avoid Code-Base violation (modeled as Soft constraint)

Fitness priority/weight consideration for fitness function integration:

- ***Project Make-Span (Overall)*** & ***Average Project Make-Span (Individual)*** are considered equally important, thus Medium.
- ***Priority-Weighted Plan-Lead*** is good to be minimized, but not as critical as shortening individual and overall project make-span, thus we give Low weight. We normalize the weight to 1 corresponding to the calculated average project priority, which is based on project cost. (The actual computation formula is shown in ***Section 2.4 Fitness Function Formulas***.)
- ***Code Base Violation*** is hard constraint modeled as soft constraint, we give High weight: x50

Fitness Breakdown	Fitness Priority	Initial Values for All Controlled GA Runs during Evaluation	
Proj Make-Span (Overall)	Medium	492.89	<- To shorten entire projects span (ALL projects)
Average Proj Make-Span (Individual)	Medium	216.73	<- To shorten average individual project span
Priority-Weighted Plan-Lead	Low	43.22	<- To schedule important projects to start earlier
Code-Base Violation x Weight	High	15195.33	<- Heuristic Weight (Hard -> Soft constraint): **50**
Integrated Fitness - To Min()	N.A.	**15948.18**	<- To Minimize

Our fitness/model design bias toward firstly satisfying all Hard and Soft constraints before focusing on shortening the Make-Spans.

2.4. Fitness Function Formula

- Project Make-Span (Overall)
  Formula: ***Value = MAX(All Projects' Prod End-Date) - MIN(All Projects' Plan Start-Date)***

- Average Project Make-Span (Individual)
  Formula: ***Value = Average (Each Project's Prod End-Date - its Plan Start-Date)***

- Priority-Weighted Plan-Lead
  Formula: ***Value = Average (Each Project's Plan Lead x Project Priority x Priority Factor)***
  Each Project's Project Priority = Each Project's cost / Total Project cost
  Project Priority Factor = 1 / Average (Each Project's Project Priority)

- Code Base Violation
  For a given testing environment, i.e. SIT, we consider the violation true, when the schedules (based on Start-Date and End-Date) of projects with different Code Base versions overlap with each other.
  We count the number of projects having violations, as well as the total overlapping days of violations for each applicable testing environment.
  Formula:
  Violation Count:
  Value = Sum of the time of violations between 2 selected projects permutation
  Violation is TRUE when: ***IF Project-A's End-Date > Project-B's Start-Date &***
  Project-B's End-Date > Project-A's Start-Date &
  Project-A's Code-Base <> Project-B's Code-Base

  Violation Days (simplified approximation):
  When Violation is TRUE, Value = Sum of the overlapping days of all projects with Code-Base violation
  Simplified overlapping between Project-A and Project-B:
  MIN (Project-A's End-Date - Project-B's Start-Date,
  Project-B's End-Date - Project-A's Start-Date)

- Integrated Fitness
  Formula: Value = ***Project Make-Span (Overall) +***
  Average Project Make-Span (Individual) +
  Priority-Weighted Plan-Lead +
  Code-Base Violation x 50 (Weight)

2.5. Constraint

Major constraint sources: (1) ***Maximum available FTE*** (2) ***Lead (Days)*** (3) ***Code Base version conflict***. Although all are Hard constraint in reality, we model first two constraints as Hard, and the third as Soft.

- Hard Constraints
  (1) Total Available Resource / FTE
  Total number of available Knowledge Engineers in this model is capped at 8.
  Total number of available Software Engineers in this model is capped at 24.
  For each project, we also define the minimum FTE (KE/SE) at 0.25 (1/4), and maximum FTE at 5. Practical reasons:
  One resource can be assigned to max 4 projects;
  Any project can be assigned max 5 FTE for each type of recourse.
  (2) Lead (Days): 0 <= Lead <= 365 (Integer)

- Soft Constraint
  (3) Code Base Violation
  The reason for modeling Code Base violation as Soft is for the purpose of generating more 'valid' solutions in the initial GA run, as well as providing more guiding for GA search by factoring in Code Base 'Violation Days' instead of 'Violation Count' into fitness function.

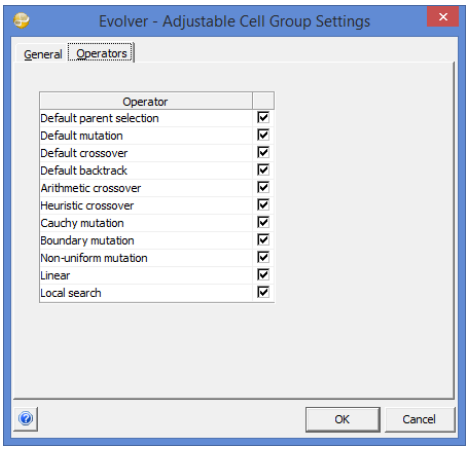
2.6. Genetic Operators

We used Evolver's ***Recipe solving method*** for Resource/FTE and Project Lead Days permutation, with:

- Default Crossover Rate: 50%
  Evolver manual mentions that 50% means about half of the genes come from either parent. Adjusting Crossover Rate can bias genetic inheritance toward one parent. (And Evolver's default Selection Method is 'Rank'.)
  Adjusting this rate can be suitable to our problem after GA reaches initial premature convergence, when the best solution still has Code-Base Violation. We can adaptively increase the rate to preserve the majority genes of best solutions, while attempting biased crossover to obtain small portion of genes from other solutions/parents, in order to jump out of the Code-Base Violation stagnation/trap.

- Mutation Rate: 50% and Population Size: 500

  We set high mutation and population is because of the very large and complex search space. Initial GA runs (crossover) can hardly generate valid solutions without Code-Base Violation, and GA normally rapidly converges prematurely (within minutes). With high mutation and population, and allowing GA to run long enough (several hours) can most likely reach solutions without (heavily penalized) Code-Base Violation in the end.



When using the ***Recipe solving method***, any number of above operators may be selected. When multiple selections are made, Evolver will test valid combinations of the selected operators to identify the best performing ones for your model. After a run, the ***Optimization summary worksheet*** ranks each of the selected operators by their performance during the run. For subsequent runs of the same model, selecting just the top performing operators may lead to faster, better performing optimizations. [2]

- Adaptive GA

  We'd like to investigate below two questions in our model. (To be discussed further in ***Section 3 Evaluation and Results***)

  (1) Weather iteratively switching Mutation Rate from 50% to 90% after premature convergence, and from 90% back to 50% after jumping out of local optima, can help?

  (2) Weather iteratively switching Crossover Rate from 50% to 90% after premature convergence, and from 90% back to 50% after jumping out of local optima, can help?

- Other Consideration

  The better operators for our problem may be a combination of:

  For ***FTE*** permutation : ***Budget***

  For ***Lead*** permutation : ***Partially Mapped Crossover***, which preserves the partial schedule sequences from both parents as much as possible.

  We also tried ***Recipe*** & ***Budget*** hybrid method, separating ***Lead*** and ***FTE*** permutation. The overall results were surprisingly **worse** than using ***Recipe*** method alone for entire chromosome. More details are discussed in ***Section 3.5. Other Evaluations - Budget vs. Recipe Solving Method for FTE Permutation***.

3. Evaluation and Results

We evaluate the GA model in two main approaches, with same initial fitness (project data inputs) for all GA runs:

- Conduct Fixed, Mutation-Adaptive and Crossover-Adaptive GA runs for ten rounds respectively, at 5 minutes per GA run. Thus in total there are **30** GA runs, covering **150 minutes** or **2.5 hours** running time.

- Conduct one round of GA run with respective longer time limits: 10, 20, 30, 40, 50, 60 minutes, and 2, 4, 6, 8 hours. Thus in total there are **10** GA runs, covering **1,360 minutes** or **22.67 hours** running time.

We adopted Evolver's default Selection Method, Generation Gap and other GA parameters.

With collected evaluation statistics above, we then visually plot and compare the evaluation results.

3.1. Theoretically Perfect Project Make-Span Benchmark

Based on known facts described in ***Section 2.1 Background Knowledge and User Configurable Inputs*** and assumption that all projects can run concurrently, we can roughly calculate theoretically perfect project make-spans using 'Total Project Efforts' and 'Maximum Available FTEs', **without** considering hard and soft constraints. The ***Average Perfect Project Span*** shown in **red** in below table can serve as reference for result comparison.

Note: *The Perfect Project Spans can dynamically change according to FTE permutation during GA run. We take the final After-GA-Run ' Perfect Project Span' as benchmark for further comparison against optimized '(Average) Proj Make-Span (Individual)' in fitness function.*

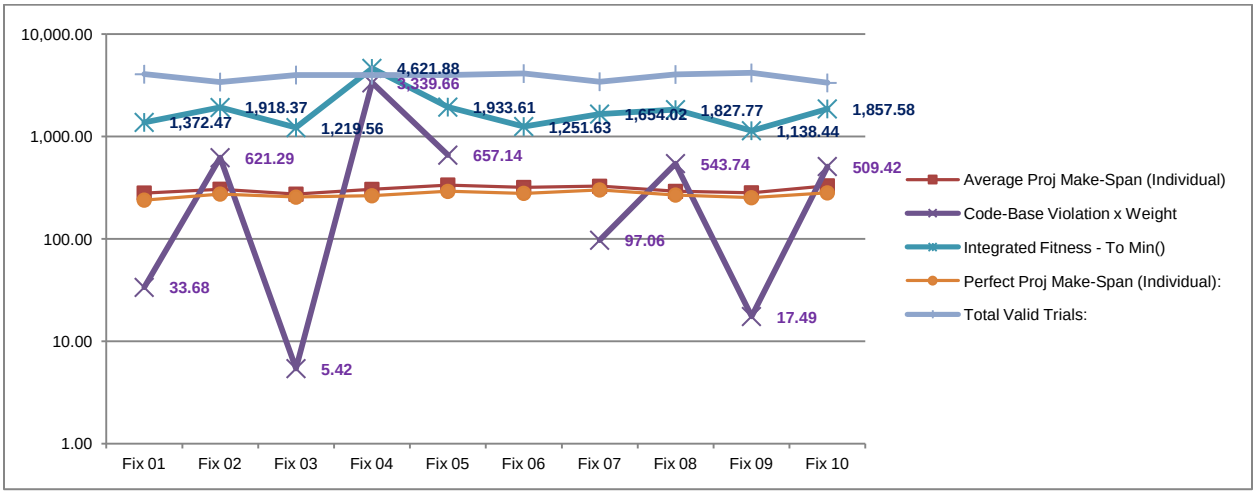
	Benchmark Reference (Theoretically Perfect Project Make-Span without constraints)		
Project Name	**Perfect Project Span - KE (Calendar-Days)**	**Perfect Project Span - SE (Calendar-Days)**	**Perfect Project Span (Calendar-Days)** =Max(Span-KE,Span-SE)
Project-A01	74.80	108.53	108.53
Project-A02	66.00	117.33	117.33
Project-A03	70.40	39.60	70.40
Project-A04	154.00	26.40	154.00
Project-A05	26.40	117.33	117.33
Project-A06	228.80	64.53	228.80
Project-A07	409.20	121.73	409.20
Project-A08	22.00	16.13	22.00
Project-A09	215.60	126.13	215.60
Project-A10	158.40	127.60	158.40
Project-A11	277.20	117.33	277.20
Project-A12	286.00	133.47	286.00
Project-A13	422.40	44.00	422.40
Project-A14	171.60	58.30	171.60
Project-A15	158.40	108.90	158.40
Average	**182.75**	**88.49**	**194.48**
Total	**2,741.20**	**1,327.33**	**2,917.20**

3.2. Controlled Evaluation - 5 minutes for each GA run, with 3 different GA settings

For two Adaptive GA settings, we manually decrease and increase Mutation/Crossover Rate from 90% to 50% (and 50% to 90%) roughly every 1 minute, thus keeping 50% or 90% setting running for one minute. There are around 9 flips during each 5 minutes GA run.

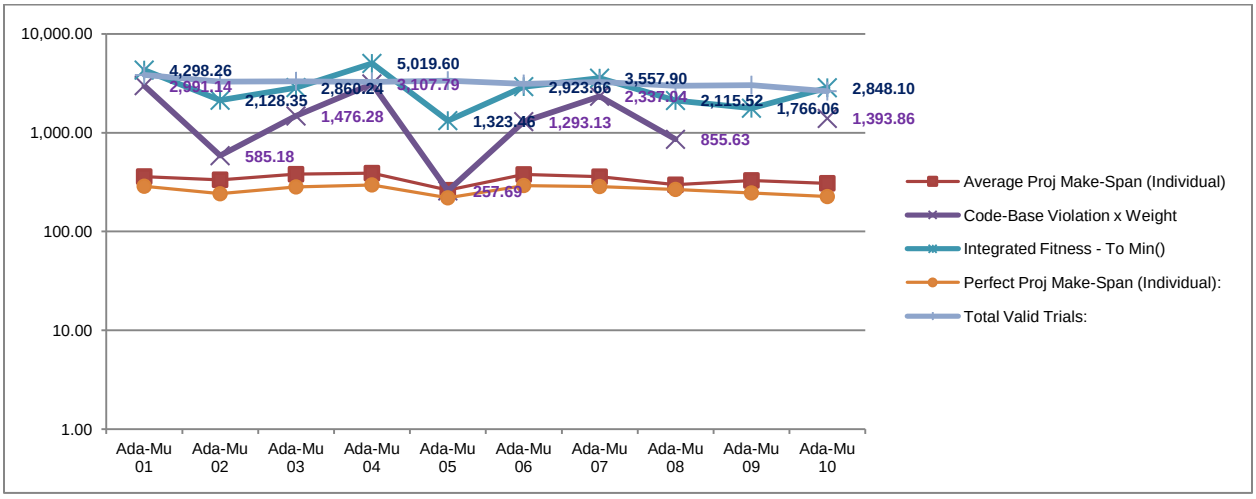
- ❖ Statistics and Plot of the Fixed GA runs

(1) Run 10 GA with 'Fixed' settings: Crossover 50%, Mutation 50%, Population 500	Fix 01	Fix 02	Fix 03	Fix 04	Fix 05	Fix 06	Fix 07	Fix 08	Fix 09	Fix 10
Proj Make-Span (Overall)	968.43	934.20	881.42	939.76	905.04	889.08	1,114.57	900.78	792.39	961.58
Average Proj Make-Span (Individual)	280.53	305.43	273.43	304.44	335.68	317.79	326.47	291.07	281.84	329.75
Priority-Weighted Plan-Lead	89.83	57.46	59.29	38.02	35.75	44.76	115.91	92.18	46.73	56.83
Code-Base Violation x **Weight**	**33.68**	**621.29**	**5.42**	**3,339.66**	**657.14**	**0.00**	**97.06**	**543.74**	**17.49**	**509.42**
Integrated Fitness - To Min()	**1,372.47**	**1,918.37**	**1,219.56**	**4,621.88**	**1,933.61**	**1,251.63**	**1,654.02**	**1,827.77**	**1,138.44**	**1,857.58**
Perfect Proj Make-Span (Individual):	239.29	274.09	255.86	263.81	291.09	278.65	300.55	268.09	252.47	281.39
Total Valid Trials:	4,050	3,398	3,978	3,963	3,985	4,107	3,424	4,024	4,170	3,332
Total Trials:	65,540	65,283	63,523	65,259	66,318	63,297	55,860	64,549	63,040	54,226
Code Base Constraint (Satisfied): 1	0	0	0	0	0	1	0	0	0	0
Code Base Constraint (Unsatisfied): 0	1	1	1	1	1	0	1	1	1	1



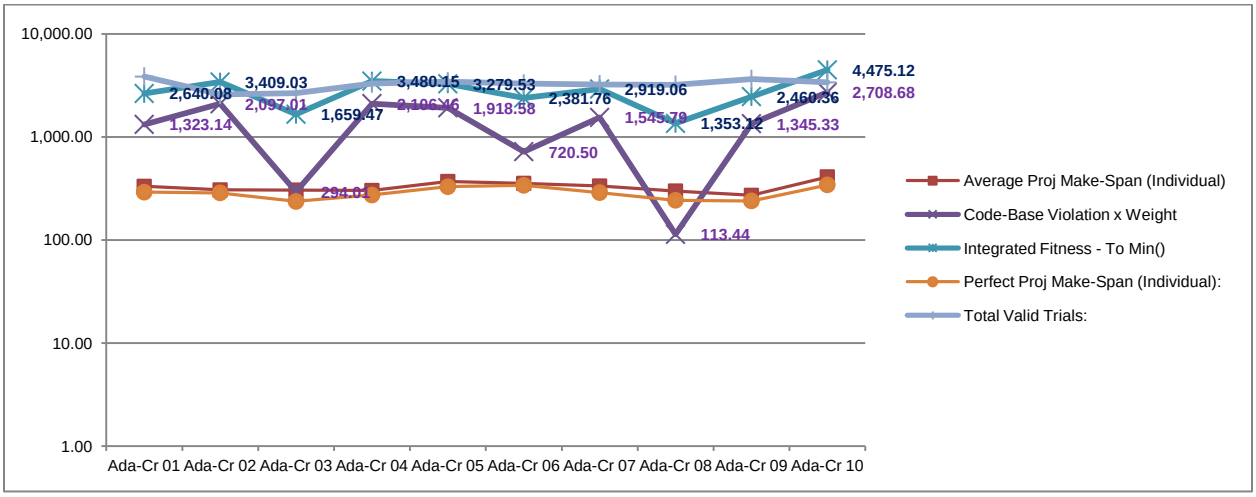
❖ Statistics and Plot of the Adaptive-Mutation GA runs

(2) Run 10 GA with manually 'Adaptive Crossover' settings: Crossover 50%, Mutation between 50%-90%, Population 500	Ada-Mu 01	Ada-Mu 02	Ada-Mu 03	Ada-Mu 04	Ada-Mu 05	Ada-Mu 06	Ada-Mu 07	Ada-Mu 08	Ada-Mu 09	Ada-Mu 10
Proj Make-Span (Overall)	908.72	1,012.97	968.27	1,450.91	731.17	1,172.02	827.44	903.70	1,400.75	1,036.00
Average Proj Make-Span (Individual)	359.68	333.59	379.60	389.35	261.58	377.69	358.16	296.75	328.49	307.46
Priority-Weighted Plan-Lead	38.72	196.61	36.09	71.54	73.02	80.81	35.26	59.44	36.81	110.77
Code-Base Violation x **Weight**	2,991.14	585.18	1,476.28	3,107.79	257.69	1,293.13	2,337.04	855.63	0.00	1,393.86
Integrated Fitness - To Min()	4,298.26	2,128.35	2,860.24	5,019.60	1,323.46	2,923.66	3,557.90	2,115.52	1,766.06	2,848.10
Perfect Proj Make-Span (Individual):	287.05	240.97	282.46	296.04	219.10	292.00	285.20	266.50	245.71	225.72
Total Valid Trials:	3,830	3,274	3,315	3,248	3,362	3,110	3,257	2,991	3,028	2,629
Total Trials:	65,693	51,876	54,684	51,613	51,811	49,382	54,241	51,128	49,239	41,173
Code Base Constraint (Satisfied): 1	0	0	0	0	0	0	0	0	1	0
Code Base Constraint (Unsatisfied): 0	1	1	1	1	1	1	1	1	0	1



❖ Statistics and Plot of the Adaptive-Crossover GA runs

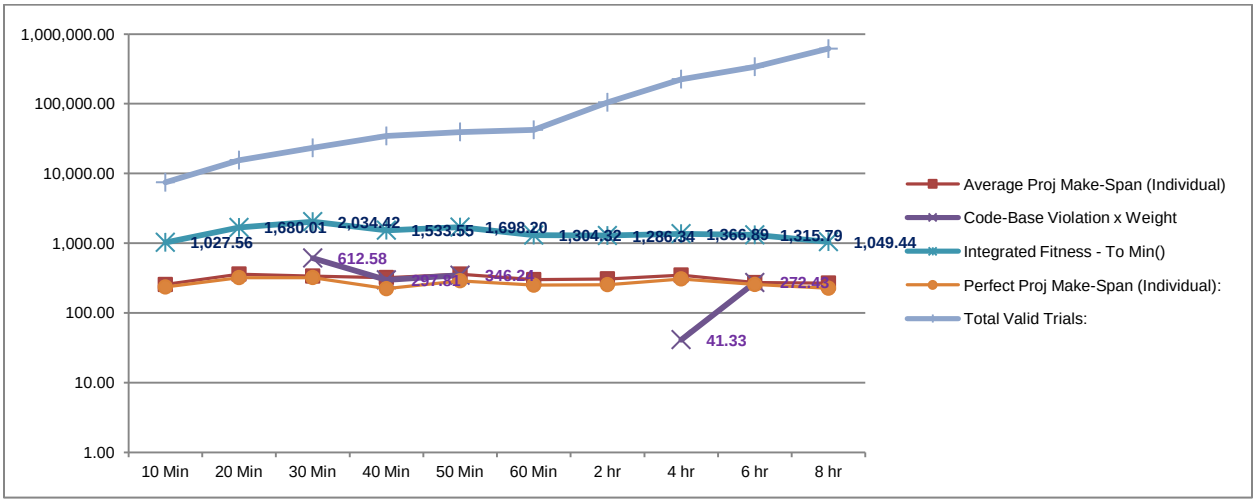
(3) Run 10 GA with manually 'Adaptive Mutation' settings: Crossover between 50%-90%, Mutation 50%, Population 500	Ada-Cr 01	Ada-Cr 02	Ada-Cr 03	Ada-Cr 04	Ada-Cr 05	Ada-Cr 06	Ada-Cr 07	Ada-Cr 08	Ada-Cr 09	Ada-Cr 10
Proj Make-Span (Overall)	953.89	943.82	966.14	979.18	947.21	1,234.09	932.59	892.92	786.04	1,319.20
Average Proj Make-Span (Individual)	331.38	306.05	304.64	302.01	369.17	354.35	333.71	298.26	271.00	409.19
Priority-Weighted Plan-Lead	31.67	62.16	94.68	92.50	44.57	72.82	106.97	48.50	57.99	38.04
Code-Base Violation x **Weight**	1,323.14	2,097.01	294.01	2,106.46	1,918.58	720.50	1,545.79	113.44	1,345.33	2,708.68
Integrated Fitness - To Min()	2,640.08	3,409.03	1,659.47	3,480.15	3,279.53	2,381.76	2,919.06	1,353.12	2,460.36	4,475.12
Perfect Proj Make-Span (Individual):	290.94	286.56	236.32	272.95	328.57	338.93	288.12	242.73	239.51	342.58
Total Valid Trials:	3,847	2,574	2,654	3,304	3,448	3,288	3,235	3,199	3,634	3,380
Total Trials:	63,035	40,666	40,209	52,304	54,758	51,957	51,801	51,850	54,875	54,577
Code Base Constraint (Satisfied): 1	0	0	0	0	0	0	0	0	0	0
Code Base Constraint (Unsatisfied): 0	1	1	1	1	1	1	1	1	1	1



3.3. Controlled Evaluation - GA with longer time limits

❖ Statistics and Plot of the Longer Time Limits

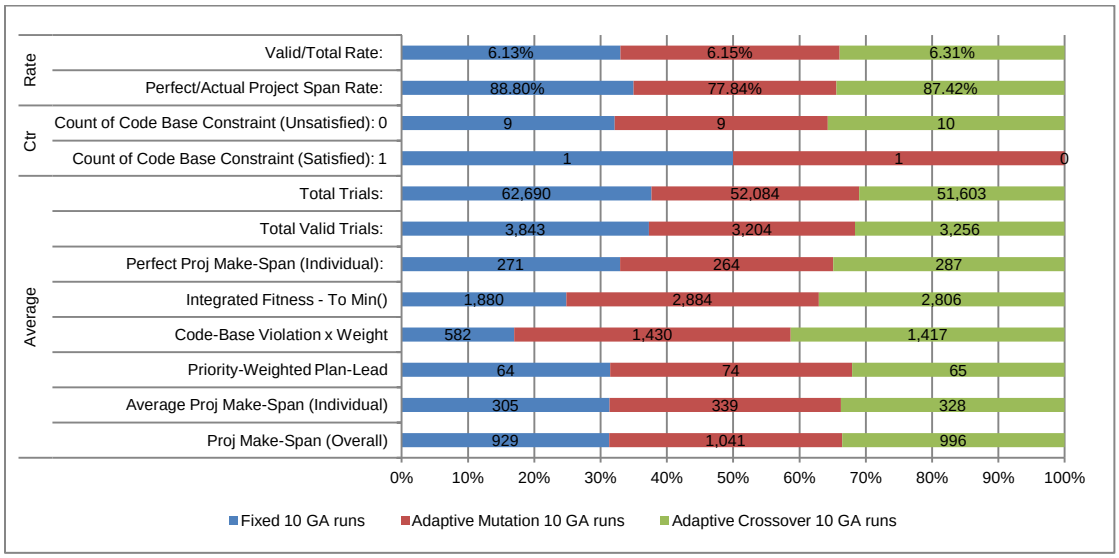
(4) Adhoc GA runs with Longer Time Limits	10 Min	20 Min	30 Min	40 Min	50 Min	60 Min	2 hr	4 hr	6 hr	8 hr
Proj Make-Span (Overall)	725.89	1,231.76	1,017.15	858.02	895.73	943.67	889.71	918.30	728.92	712.46
Average Proj Make-Span (Individual)	254.98	360.56	337.11	321.16	359.89	299.81	307.72	348.69	271.69	267.75
Priority-Weighted Plan-Lead	46.69	87.68	67.58	56.56	96.35	60.84	88.90	58.58	42.75	69.24
Code-Base Violation x **Weight**	**0.00**	**0.00**	**612.58**	**297.81**	**346.24**	**0.00**	**0.00**	**41.33**	**272.43**	**0.00**
Integrated Fitness - To Min()	**1,027.56**	**1,680.01**	**2,034.42**	**1,533.55**	**1,698.20**	**1,304.32**	**1,286.34**	**1,366.89**	**1,315.79**	**1,049.44**
Perfect Proj Make-Span (Individual):	235.06	320.68	321.42	222.20	288.98	249.63	254.05	306.96	255.45	225.71
Total Valid Trials:	7,465	15,530	23,296	34,498	39,417	42,250	104,758	225,310	339,664	616,509
Total Trials:	118,943	266,408	400,074	561,554	614,968	690,031	1,648,972	3,588,055	5,262,480	7,066,485
Code Base Constraint (Satisfied): 1	1	1	0	0	0	1	1	0	0	1
Code Base Constraint (Unsatisfied): 0	0	0	1	1	1	0	0	1	1	0



3.4. Overall Results Visualization and Comparison

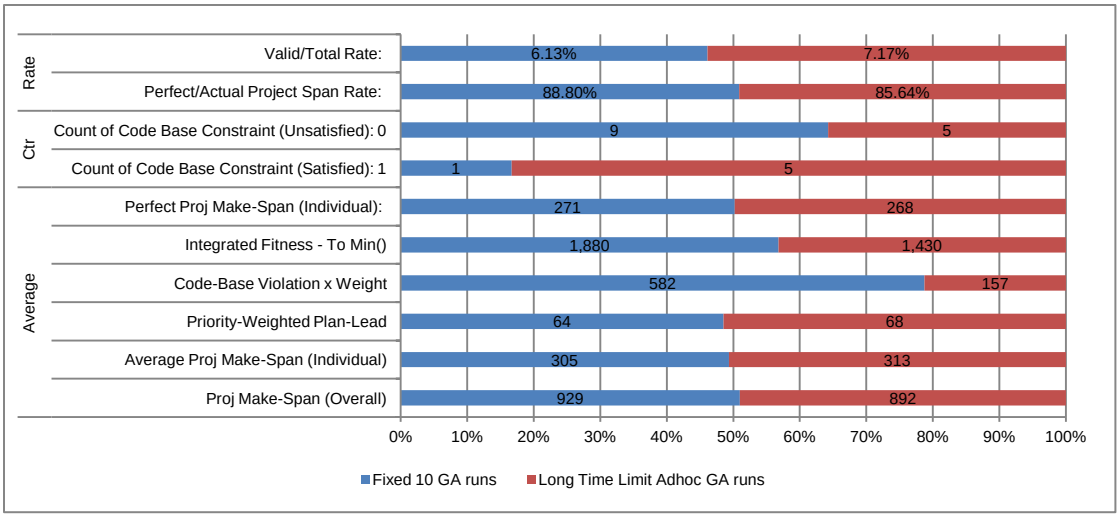
❖ Plot of **Averages** among the Fixed, Adaptive-Mutation and Adaptive-Crossover

		Fixed 10 GA runs	Adaptive Mutation 10 GA runs	Adaptive Crossover 10 GA runs
Average	Proj Make-Span (Overall)	929	1,041	996
	Average Proj Make-Span (Individual)	305	339	328
	Priority-Weighted Plan-Lead	64	74	65
	Code-Base Violation x **Weight**	**582**	**1,430**	**1,417**
	Code-Base Violation (**Days**)	**11.65**	**28.60**	**28.35**
	Integrated Fitness - To Min()	**1,880**	**2,884**	**2,806**
	Perfect Proj Make-Span (Individual):	271	264	287
	Total Valid Trials:	3,843	3,204	3,256
	Total Trials:	62,690	52,084	51,603
Ctr	Count of Code Base Constraint (Satisfied): 1	1	1	0
	Count of Code Base Constraint (Unsatisfied): 0	9	9	10
Rate	Perfect/Actual Project Span Rate:	88.80%	77.84%	87.42%
	Valid/Total Rate:	6.13%	6.15%	6.31%



❖ Plot of **Averages** between the Fixed and the Longer Time Limits

		Fixed 10 GA runs	Long Time Limit Adhoc GA runs
Average	Proj Make-Span (Overall)	929	892
	Average Proj Make-Span (Individual)	305	313
	Priority-Weighted Plan-Lead	64	68
	Code-Base Violation x **Weight**	**582**	**157**
	Code-Base Violation (**Days**)	**11.65**	**3.14**
	Integrated Fitness - To Min()	**1,880**	**1,430**
	Perfect Proj Make-Span (Individual):	271	268
	Total Valid Trials:	3,843	144,870
	Total Trials:	62,690	2,021,797
Ctr	Count of Code Base Constraint (Satisfied): 1	1	5
	Count of Code Base Constraint (Unsatisfied): 0	9	5
Rate	Perfect/Actual Project Span Rate:	88.80%	85.64%
	Valid/Total Rate:	6.13%	7.17%



Results shows:

(1) Shorter GA run time (5 minutes) is subject to more fluctuation of fitness values, compared with longer run time. Shorter GA run Code-Base constraint satisfied at **10%**.

(2) ***Code-Base Violation x Weight*** is the major contributor to fluctuation of fitness values. Because similar trend is observed for Code-Base and Overall integrated fitness.

(3) Longer GA run shows statistical fitness improvement, especially the occurrences of satisfying Code-Base violation constraint. Longer GA run Code-Base constraint satisfied at **50%**.

(4) Fluctuation of fitness values: ***Proj Make-Span (Overall)*** & ***Average Proj Make-Span (Individual)*** is not relatively strong (flat) among different GA runs.

(5) ***Average Proj Make-Span (Individual)*** is close to (above 85%) ***Theoretically Perfect Proj Make-Span***. Thus it's good.

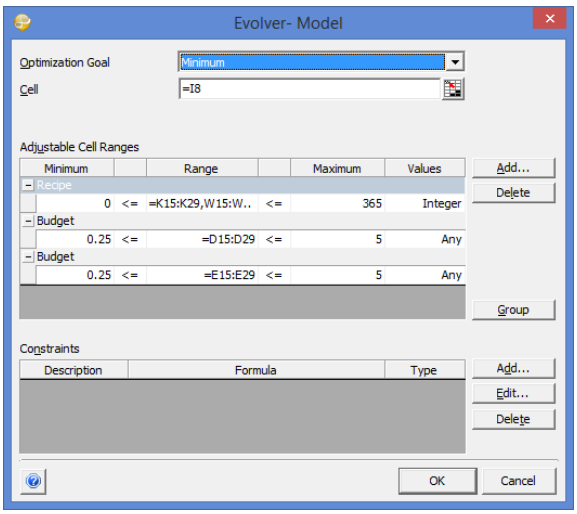
(6) Adaptive GAs don't improve performance in this context.

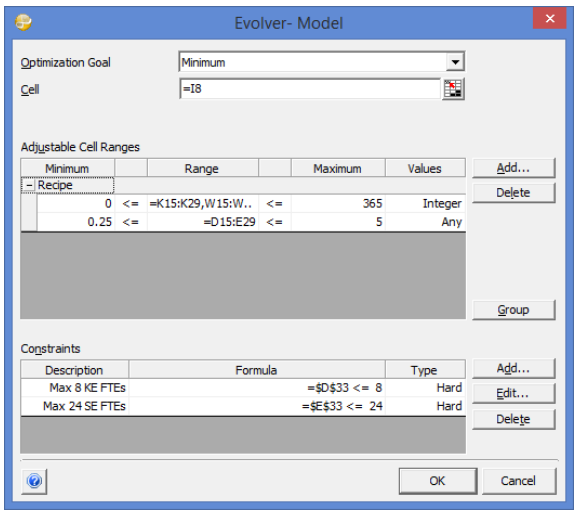
(7) The best solution obtained is the longest GA run (8 hours) in our evaluation, followed the 2nd best GA run (10 minutes).

(8) The Average Code-Base Violation (Days) is **11.65** and **3.14** respectively for Fixed and Long running GA. ***The extra few days delay can normally be tolerated in real project planning.***

3.5. Other Evaluation - Budget vs. Recipe Solving Method for FTE Permutation

- Evolver Tool Settings
  (Use Recipe solving method for both Leads Permutations.)





	Budget Method for FTE	**Recipe Method for FTE**
Valid Trails	All Trials are Valid ones.	Not all trials are Valid ones due to FTE Hard constraint. ***Valid/Total Trial Rate*** is about 6% only.
Initial Solutions Space	GA starts to effectively work immediately to improve fitness values.	Fitness values remains unchanged for around one minute, before GA start to effectively find valid solutions to work to improve fitness.
Lead Optimization	The Plan Leads and other UAT/NFT/Prod Leads are long. Considerable Leads reached '365 Days' the maximum boundary set for Lead. ***This may be a disadvantage resulted by separating GA chromosome to be solved by 2 different methods (Budget and Recipe).***	The Plan Leads seem shorter (better), and other UAT/NFT/Prod Leads are almost perfect (Majority is zero value). ***This may be an advantage resulted by using Recipe for the entire GA chromosome to be considered/solved as a whole.***
Overall Results	Worse	Better

4. Conclusion

4.1. Insights

This release scheduling problem is very challenging in real IT industry. Although Genetic Algorithm is proved to be an effective optimization tool, nevertheless even with our careful modeling, there is still high tendency for GA to stuck in local optima (for un-predictable very long time) with Code-Base soft constraint (critical resource contention) unsatisfied.

Empirically we found, in our problem setting:

- We noticed there are occasions when ***Code-Base Violation*** hard constraint can be satisfied as early as initial 4-8 minutes. If the constraint is not met in early stage, the GA run tends to rest in local optima for long (even hours), and the 500 population has nearly ***no diversity***, thus further fitness improvement relies purely on mutation to improve ***Code-Base Violation*** hard constraint, to jump out and jump farther to other search regions. This is the major reason we set 50% high mutation and 500 large population in this context.

- Since the search space seems deceptive, allowing GA to run longer for several hours is likely to satisfy all constraints to achieve acceptable good solutions. The trend of better fitness and Code-Base constraint is observed.

- To increase the successful possibility for solutions, we can use the approach of running several GA concurrently (or sequentially) for few hours, i.e. 10 x 1 hour.

GA run with several hours or overnight batch can still be considered acceptable for real IT industries. Because in current IT project and release management encompassing, it's not uncommon for a group of project owners to spend week(s) in discussion, alignment, negotiation and planning release schedules and resource allocation to accommodate all running projects across different departments. Shortening the project release optimization can save considerable man power, and to avoid potential political stalemate, via more 'objective' computational intelligence.

This basic GA model promisingly serves as a more efficient tool for management to centrally coordinate and control release planning.

4.2. Future Work

- To design and use Partially Mapped Crossover GA operator.
- To allow user to define and set Project Priority.
- To allow user to define and set Number of testing environments, which can run different Code-Bases in parallel: i.e. SIT x 3, UAT x 2.
- To allow user to set which testing environment is subject to Code-Base violation.
- To allow user to fix certain calendar dates, i.e. UAT start/end dates must be fixed due to external vendor contracts, Production Go-Live date must be fixed due to business marketing campaign or government regulation.
- To allow user to add additional/reserved FTEs (i.e. allowing certain number of potential new hires to get on board), to ease project release planning.
- To intelligently suggest Code-Base Mergers when it's possible to: (1) solve unavoidable Code-Base Violation (2) to significantly shorten project make-span.
- To allow introducing new projects to existing planned/optimized project release portfolios/plans, and to cause minimal changes to existing project schedules during new round of schedule optimization.

5. Appendix

5.1. GA Model in Excel

Project Release Optimizer.xlsx

5.2. Evaluation Statistics Visual Dashboard

Visual Dashboard.xlsx

5.3. Evaluation Statistics of Each GA Run

Evaluation Satistics.zip

6. Reference

[1] Release Management - Wikipedia
http://en.wikipedia.org/wiki/Release_management

[2] Guide to Using Evolver The Genetic Algorithm Solver for Microsoft Excel
http://www.palisade.com/downloads/manuals/EN/Evolver5_EN.pdf

[3] Junchao Xiao, Leon J. Osterweil, Qing Wang, Mingshu Li,
Dynamic Scheduling in Systems with Complex Resource Allocation Requirements
http://laser.cs.umass.edu/techreports/09-049.pdf