

**Department of Industrial Engineering and Management,
Yuan-Ze University, Taiwan, R.O.C.**

The Production Scheduling and Soft-Computing Lab

**The Simulated Annealing for Solving the
Quadratic Assignment Problem and
Continuous Problem**

1. Introduction

The work solves the quadratic assignment problem (QAP) and continuous problem by simulated annealing (SA). Besides, it also applies some features of Genetic algorithm (GA), including the encoding scheme, selection method, and mutation operator. Thus, it is a GA-Like SA. Then, we also introduce some techniques to improve the solution quality. Finally, the paper designs a called component, which is named OpenSA, to solve different kinds of problems. The work is organized as following.

The section 2 and section 3 describe how to solve QAP and continuous problem by SA respectively. Then, the study tries some techniques, such as reheating and restore solution, to enhance the solution quality of SA in section 4. Depending on above infrastructure, the study composes an object-oriented component that is written in Java and it's presented in section 5. The goal of designing the component is to reduce the complexity and easy to use when we solve different kinds of problem. Section 6 is the experimental result of the two cases and the section 6 is the discussion and conclusions.

2. Solving the QAP

No matter for solving the combinatorial problem or continuous problem, the main procedures of SA can be distinguished into generating an initial solution, moves, evaluating the new solution(s), accepting rules, and repeated iteration. The following sub-sections describe them in detail.

2.1 Generate an Initial Solution

Before generating an initial solution, we should determine how to encode the problem. As for the QAP, which is one of the combinatorial optimization problem, the sequential encoding or path representation type like {1, 2, 3...} is employed here. If the problem is the continuous problem, it may be encoded into the type of real number or binary form like binary {0,1}. Because the later part of the study aims at the continuous problem, they are discussed later. After we decide the technique to encode the problem, the work randomly generates an initial solution for the problem which assigns each department at exactly one dimension.

In order to make the explanation more clear, the work illustrates an example by the Nug30. There are 30 departments, whose number are 0, 1, 2, ..., and 29. Then, the locations of these 30 departments are known in advance. Besides, the flow quantity between two departments is also given. The figure 2.1 presents the encoding of QAP for the 30 departments.

14	5	28	24	12	11	23
----	---	----	----	------	----	----	----

Figure 2.1 The problem representation of the QAP for 30 departments

From the figure, it implies the department 14 at the first position, department 5 at the second position, and so on. Thus, the initial solution is done after we deal with the encoding and to generate initial solution.

2.2 Moves Strategy

The purpose of move is to do a variation on current solution by local search. The move strategies include swap move, 2-opt, 3-opt, k-opt, shift move, and inverse move. The swap move is to swap two points of original path and the 2-opt is to replace original two arcs, which are not nearby and then connect two new arcs into the path. The work applies the swap move, shift move, and inverse move together. They are described below.

No matter for the shift move or inverse move, it needs to randomly generate two cut points. We may call it cut point 1 and cut point 2. For shift move, we move the cut point 2 ahead the position of the range so that it replaces the original cut point 1. Then, shifting all point forward for one space until at the end of element on cut point 2. (Because it has been moved to the place of cut point 1) Thus, the shift move is done. The figure 2.2 shows how the shift move works which supposes there are 10 departments.

The inverse move is to inverse the current position. Take figure 2.3 for instance, the original sequence between the two cut points is 9-5-3-4-8-0. After the inverse, the new sequence becomes 0-8-4-3-9-5.

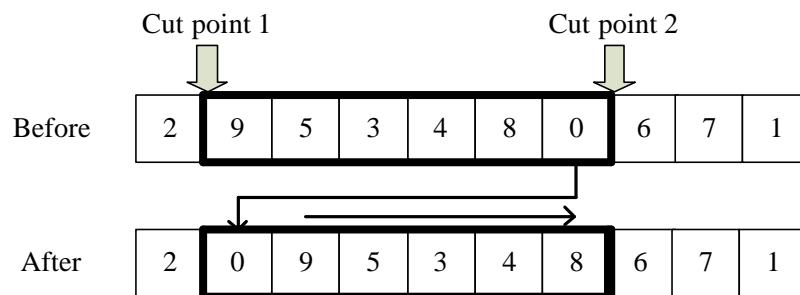


Figure 2.2 Shift move

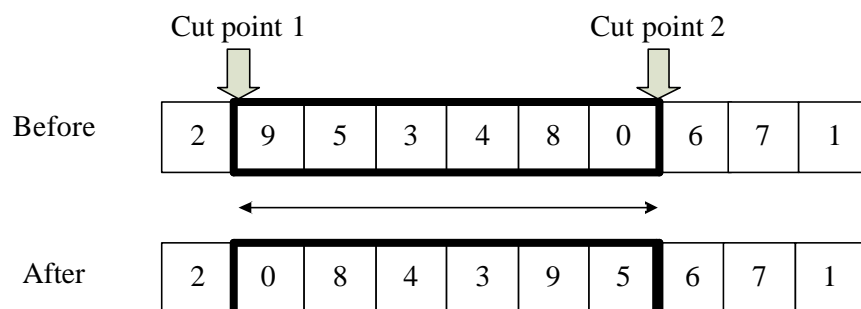


Figure 2.3 Inverse move

Finally, the swap move is very easy to implement because it just has to set two positions and exchange the two values of its position. The result is shown in the figure 2.3.

- $f(x)$: The current objective value of x .
- $f(x')$: The objective value of the neighborhood solution of x' .
- T' : The current temperature.

Then, the neighborhood solution is also compared with current best solution. If it is better than current optimal solution, SA replaces the current best solution by the neighborhood solution.

Finally, the termination condition is that when the *currentTemperature* < *finalTemperature*, the SA stops.

2.5 Additional Approaches for QAP

In the study, the standard procedures are slightly revised. For example, we introduce multiple local search operators that work in the same time and use tournament select to determine the better move at every move step. Besides, to obtain better solution quality, the research tries to use some methods, such as reheating and set back to current best technique, to do so. The following paragraph describes the modification of standard procedures and makes it become a GA-Like SA.

Because the work applies some techniques of GA, it is called a GA-Like SA. In the beginning, the solution is treated as a chromosome and each gene value corresponding to its dimension. Then, we also employ the multiple local search operators, which is a widely used mutation operator in GA. Furthermore, because the number of local search operator is more than one, it creates a small population. Since we want to select better chromosome from the population, it is natural to use the binary tournament to select better solution generated by different local search operator.

There are two different methods to enhance the solution quality. First of all, we record the number of non-improved moves. When it is up to a specific level, the algorithm does the reheating procedure. The following equation shows the operation:

$$\text{currentTemperature} = \text{currentTemperature} + \text{initialTemperature}/(\text{counter}*0.01);$$

where

currentTemperature: The current temperature

initialTemperature: The initial temperature

counter: The number of temperature changes

By the effort, the temperature is heating up. Then, the meaning of reheating procedure may mean the current solution is not good enough. Hence, after the reheating procedure, the algorithm will also set back the current solution to original current best solution. In other words, the current best solution replaces the current solution and the algorithm use it to continue the exploration.

2.6 The Main Procedures of the Algorithm

The sub section combines above efforts into one by using the pseudo code. The following is the variables defined by the study.

initialTemperature: The initial temperature

a : The changing annealing schedule
currentTemperature: The current temperature
finalTemperature: The final temperature and it is used as the termination criterion
numberOfMoves: The number of local search within the same temperature
chromosome1: The solution class to store current solution and corresponding objective values
tempChromosomes[: It's also the solution class which store temporarily solution by different kinds of moves
best: The best solution
counter: The number of *currentTemperature* is changed

Main()//the main procedures of SA for QAP problem

1. *initialParameters*()
2. *initialStage*();
3. **while** *counter* < *numberOfSolutionsExamined* **do**
4. **for** *i* = 0 to *numberOfMoves* **do**
5. *tempChromosomes* = *getMoves*();
6. *calcObjectiveValue*();
7. *tempChromosomes*[0] = *selectBetterMoves* ();
8. *acceptanceRule*();
9. *reheating*();
10. **End for**
11. *currentTemperature* *= **a** ;
12. **End while**

3. Solving the Continuous Problem

The SA procedures of continuous problem is the same when we solve the combinatorial problem in section 2 which explains the procedures of generating an initial solution, moves, evaluation of the new solution(s), acceptance rule, and repeated iteration. Of course, some of them are the same and some are different from each other. Section 3 pays attention to the differences of the QAP problem. The different places are generating an initial solution, moves, and evaluation of the new solution(s). Then, acceptance rule, and repeated iteration are identical with the section 2.4, they won't be discussed here again.

3.1 Generate an Initial Solution

The method to generate the initial solution is very easy because we only consider that the solution should locate in the boundary. The study applies the Himmelblau function as an example. The problem has two dimensions which are named x_1 and x_2 , and its corresponding boundary are lie between ± 6 . Thus, both the solution of x_1 and x_2 lie between the upper bound and lower bound is valid.

The author uses the following equation to generate each initial solution x_i .

$$x_i = \text{lwBounds}_i + U(0,1) * (\text{upBound}_i - \text{lwBounds}_i) \quad i = 1, \dots, n$$

where

n : it's the number of dimension

Therefore, if there is a random value $U(0,1) = 0.3$, then the value x_i is $-6 + 0.3*(6 - (-6)) = -2.4$. By doing that, the initial stage is done.

3.2 Moves Strategy

The move strategies of continuous problem are not as many as combinatorial problem. The research considers moving current solution up or down that is depending on random probability. If $U(0,1)$ is larger than 0.5, the current value moves up; otherwise, moves down. The moving length is depended on the range from current position to boundary. The equation of the move strategy below:

$$\begin{cases} x_i = x_i + (upBound_i - x_i) \times U(0,1) & \text{if } U(0,1) > 0.5 \\ x_i = x_i + (x_i - lwBounds_i) \times U(0,1) & \text{otherwise} \end{cases}$$

Hence, if the original x_i is -2.4 and there is a random value $U(0,1) = 0.6$ for judging the value to go up or go down, the value moves up. Besides, the other random value $U(0,1) = 0.1$, the $x_i -2.4 + (6 - (-2.4))*0.1 = -1.56$.

3.3 Evaluate the Objective Value

The Himmelblau function is a two dimension problem which shows as following:

$$Z = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7) + 0.1 \times [(x_1 - 3)^2 + (x_2 - 2)^2]$$

Suppose the x_1 is -1.56 and x_2 is 2.6 , then the objective value $Z = 35.598 + 3.24 + 2.115 = 40.95$.

4. An Object-Oriented Component Design

4.1 Introduction of the Component and Its Interfaces

The study does a well-designed callable component, which is named OpenSA. The OpenSA defines some general interfaces for each procedure of SA. Basically, the interface looks like a blue print. When the class program implements it, the behaviors defined by the interface show on the class program that is expected. The following table shows these procedures' corresponding interfaces.

Table 4.1 The purposes and its corresponding interfaces

Purposes	Interface
Control the main procedures of SA	MainI
Move strategy (Local search)	MoveI
Evaluation of objective function	ObjectiveFunctionI
Select better neighborhood solution from moves	SelectI

Take the mainI for example, it is an interface which defines the behavior of main procedures, such as starting SA, initial stage of SA, moving strategy, calculating objective values, acceptance rules, and so on. Furthermore, the reheating technique is

also integrated into the framework. Then, if we want to add others method, we can simply add it to the additional method. The figure 4.1 shows the structure of MainI which is presented by UML diagram. It also describes the there are two classes, SingleThreadSA and MainContinuous, implement the Main. Moreover, there are two applications will call the interface, QAP_NVR and Himmelblau. The MoveI, ObjectiveFunctionI, and SelectI of UML diagram shows at figure 4.2, 4.3, and 4.4 respectively.

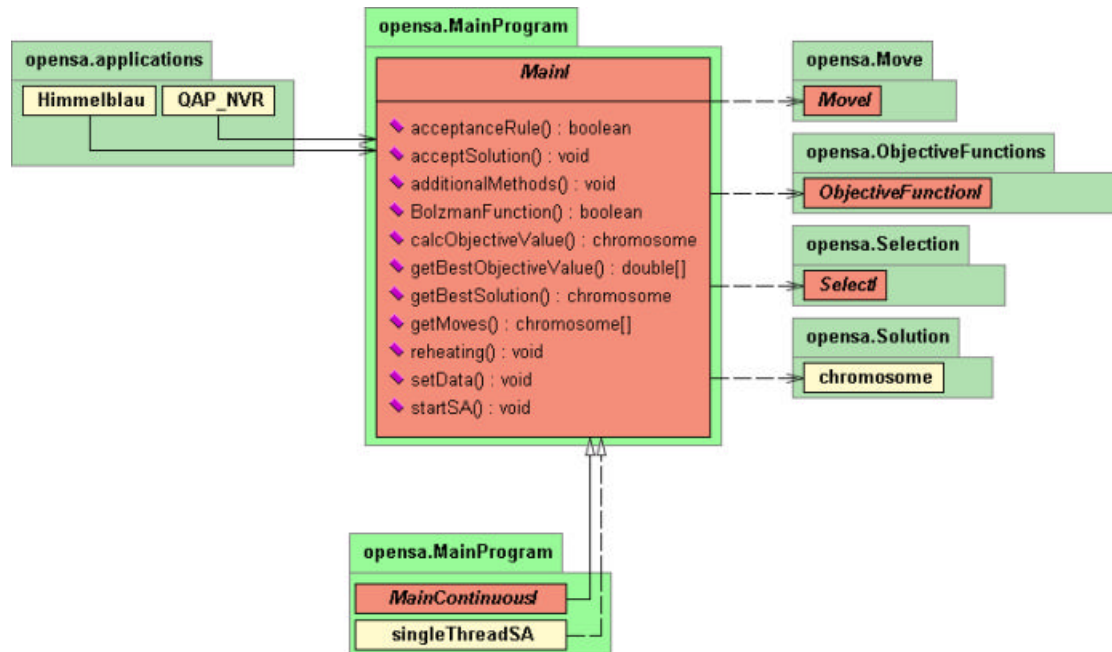


Figure 4.1 The UML diagram of MainI

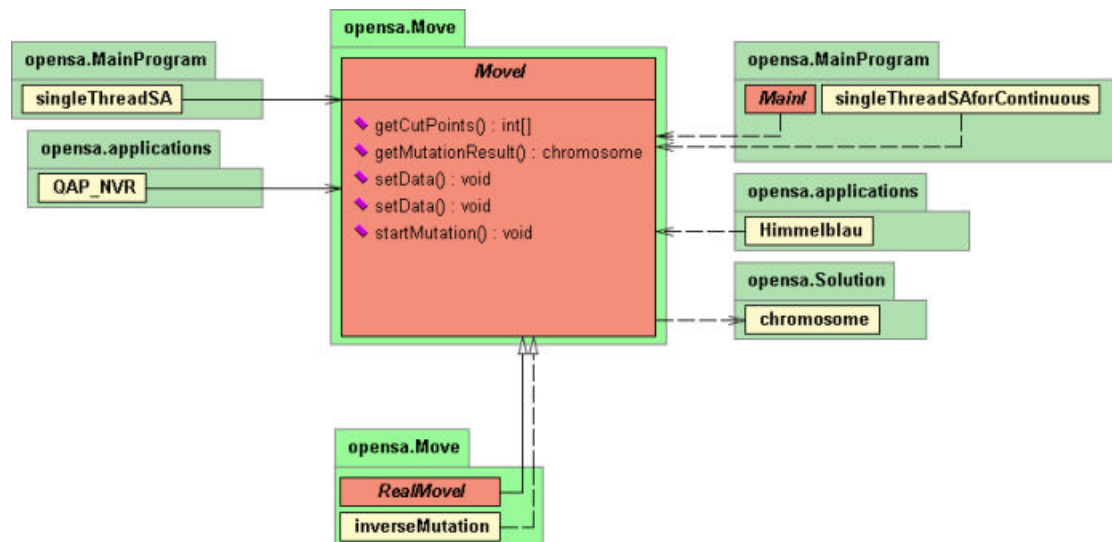


Figure 4.2 The UML diagram of MoveI

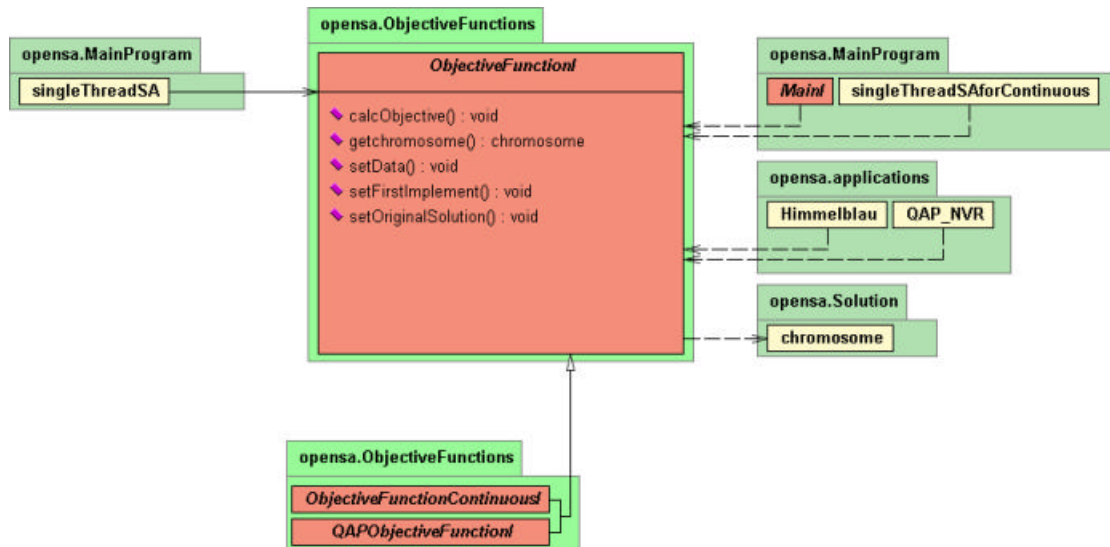


Figure 4.3 The UML diagram of ObjectiveFunctionI

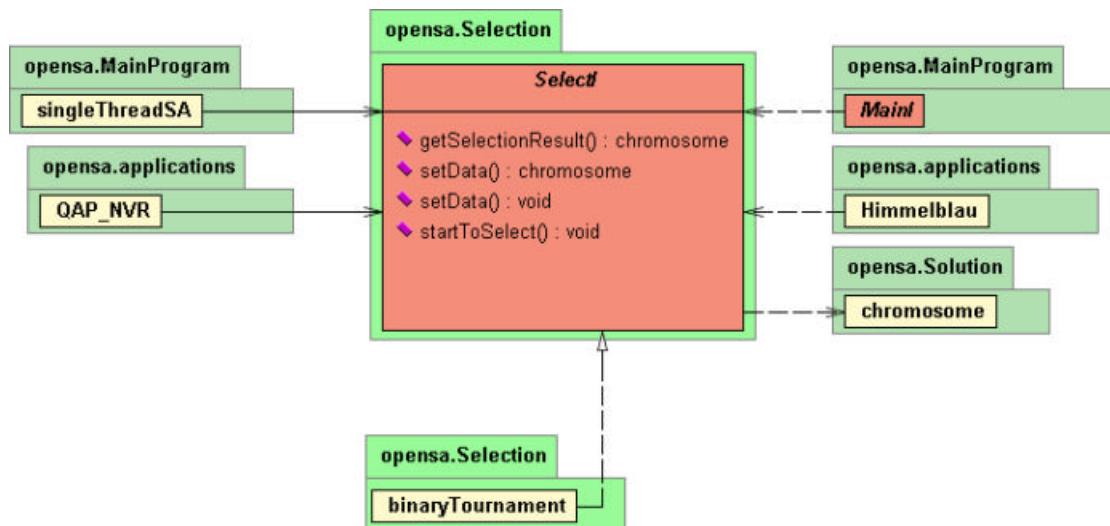


Figure 4.4 The UML diagram of SelectI

Then, in order to fit different kinds of problem, we may extend original interface and add some functions to the new interface. For example, the study is to solve combinatorial problem and continuous problem, so the input parameter for the objective function will be different. As for continuous problem concerned, it just need the value of each dimension. However, as for the QAP problem concerned, it not only has to pass the sequence data to the objective and then evaluate the solution, but also need the flow quantity and the distance between two departments. So different applications would have different requirements. It's the reason why it needs to modify original interface. The figure 4.5 demos the `QAPObjectiveFunctionI` extends the original interface `ObjectiveFunctionI` and adds a method, `setQAPData`, for the use of QAP objective function. (You can ignore the `cutPoints` method since we won't use it in the case). Finally, the class `QAPObjectiveFunction` will implements the `QAPObjectiveFunctionI`.

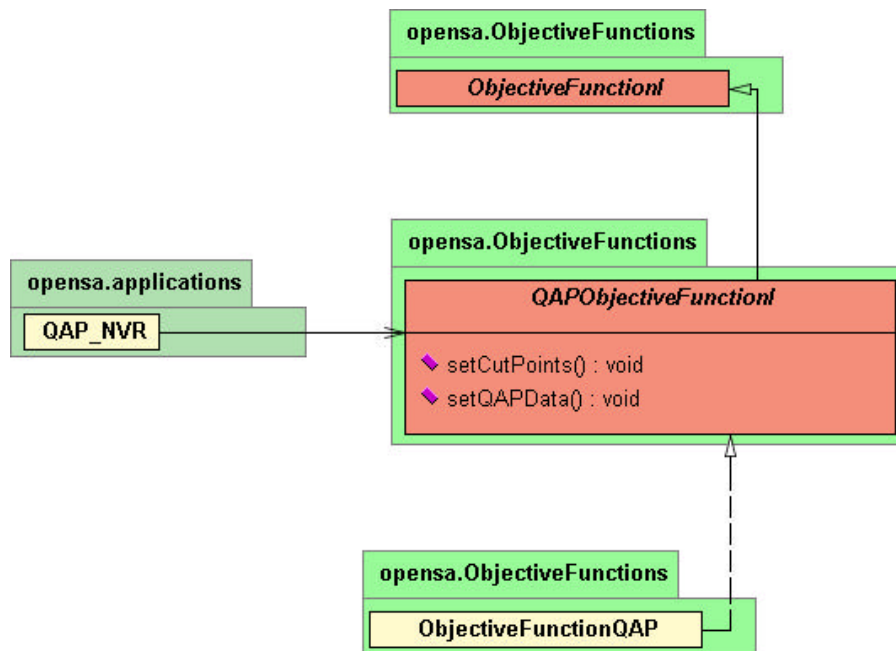


Figure 4.5 The UML diagram of QAPObjectiveFunctionI

4.2 Classes of the Component

Although the interfaces define the expected behavior of the object, it can't do anything because there is no code in the interface. However, the class programs will implement these interfaces and execute specific actions. The table 4.2 points out the purposes of different classes and the interfaces that they implement.

Table 4.2 The classes in OpenSA

Purpose	Class name	Implements the interface
The main procedures of SA when solving the combinatorial problem	singleThreadSA	MainI
The main procedures of SA when solving the continuous problem	singleThreadSAforContinuous	MainContinuousI and extends the singleThreadSA
Swap move	swapMutation	MoveI
Shift move	shiftMutation	MoveI
Inverse move	inverseMutation	MoveI
Move for real number	RealValueMutation	RealMoveI
Evaluate QAP objective value	ObjectiveFunctionQAP	QAPObjectiveFunctionI
Evaluate continuous objective value	ObjectiveFunctionContinuous	ObjectiveFunctionContinuousI
Select better moves	binaryTournament	SelectI

Take the singleThreadSA for instance, the class implements the solution MainI, so the methods (startSA, initialStage, moveStrategy...) defined by MainI are all implemented at singleThreadSA. The figure 4.6 shows the UML diagram of singleThreadSA.

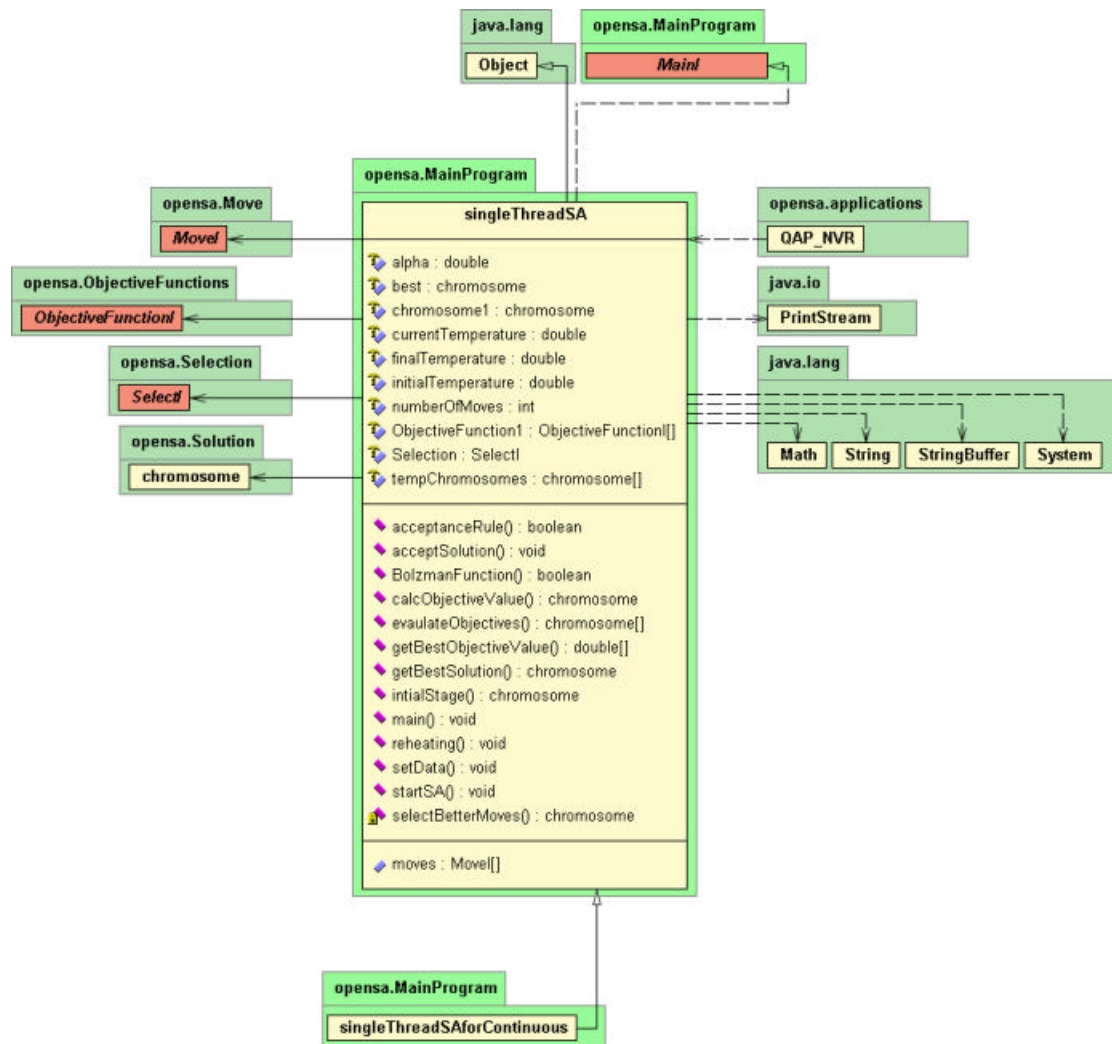


Figure 4.6 The UML diagram of singleThreadSA

5. Experimental Result

The experimental result includes the QAP problem of Nug30 and the continuous problem of Himmelblau function on P4 2.8 GHZ. The changing of annealing schedule (\mathbf{a}), changing stopping criterion, changing the initial starting point, and changing the random number seed are considered in both cases. Moreover, the work uses the ANOVA to analyze the experiment result.

The number of movements is set to 1500. The \mathbf{a} is {0.9, 0.8, 0.7}. The stopping criterion depends on the finalTemperature, which is {10, 100, 500}. Then, because the initial solution is generated randomly, it is changed with number of random seed. In other words, if we change the random seed, the initial solution changed together. The number of random seed is changed ten times here. (Other random seed produces the number of random seed).

From above description, there are three main factors. To simplicity, they are named as follows:

X: The effect from the changing of annealing schedule whose treatments are 3

Y: The effect from the changing stopping criterion whose treatments are 3

Z: The effect from the changing random seed whose treatments are 10

Hence, the Statistics model can be represented as following:

$$T = X + Y + X Y + Z + X Z + Y Z$$

Where

T: The objective value for QAP or Himmelblau function

From the Statistics model shows we consider the interaction between two factors. However, since the probability of interaction among the three factors is rare, it is not taken into consideration. Finally, the Statistics hypothesis is as following for the three

The section 5.1 show the former one and the latter one experiment result respectively.

5.1 QAP Experimental Result and ANOVA of the Experiment

The experiment result of QAP shows at the table 5.1 which has 90 combinations.

Table 5.1 The experimental result of QAP experiment

Num	a	finalTemperature	Random seed	Time*	Obj value (fij*Dij)
1	0.9	10	1578708367	81828	3197
2	0.9	10	1578708367	59562	3269
3	0.9	10	1001241461	38562	3251
4	0.9	10	1119387339	41047	3203
5	0.9	10	814391786	45531	3193
6	0.9	10	152123184	64359	3275
7	0.9	10	912023484	58672	3278
8	0.9	10	251167074	44563	3215
9	0.9	10	703090523	48094	3228
10	0.9	10	399287411	36906	3177
11	0.9	100	1578708367	32156	3224
12	0.9	100	1001241461	50031	3277
13	0.9	100	1119387339	44204	3213
14	0.9	100	814391786	38468	3343
15	0.9	100	152123184	51532	3260
16	0.9	100	912023484	36171	3190

17	0.9	100	251167074	36500	3235
18	0.9	100	703090523	51359	3187
19	0.9	100	733447773	46329	3238
20	0.9	100	399287411	46796	3230
21	0.9	500	1578708367	344	3592
22	0.9	500	1001241461	281	3614
23	0.9	500	1119387339	329	3639
24	0.9	500	814391786	281	3625
25	0.9	500	152123184	297	3573
26	0.9	500	912023484	281	3616
27	0.9	500	251167074	281	3595
28	0.9	500	703090523	313	3584
29	0.9	500	733447773	297	3625
30	0.9	500	399287411	312	3617
31	0.8	10	1578708367	28547	3276
32	0.8	10	1001241461	20031	3290
33	0.8	10	1119387339	24672	3220
34	0.8	10	814391786	22188	3238
35	0.8	10	152123184	26000	3262
36	0.8	10	912023484	26234	3201
37	0.8	10	251167074	17687	3245
38	0.8	10	703090523	28235	3282
39	0.8	10	733447773	31375	3287
40	0.8	10	399287411	27531	3246
41	0.8	100	1578708367	13734	3361
42	0.8	100	1001241461	5922	3467
43	0.8	100	1119387339	9594	3418
44	0.8	100	814391786	9375	3377

45	0.8	100	152123184	6016	3499
46	0.8	100	912023484	8141	3346
47	0.8	100	251167074	922	3585
48	0.8	100	703090523	16141	3357
49	0.8	100	733447773	5375	3510
50	0.8	100	399287411	4406	3532
51	0.8	500	1578708367	188	3657
52	0.8	500	1001241461	156	3541
53	0.8	500	1119387339	187	3681
54	0.8	500	814391786	172	3588
55	0.8	500	152123184	203	3666
56	0.8	500	912023484	188	3633
57	0.8	500	251167074	140	3661
58	0.8	500	703090523	219	3610
59	0.8	500	733447773	125	3646
60	0.8	500	399287411	188	3606
61	0.7	10	1578708367	20609	3323
62	0.7	10	1001241461	17687	3271
63	0.7	10	1119387339	18109	3253
64	0.7	10	814391786	19813	3246
65	0.7	10	152123184	13766	3268
66	0.7	10	912023484	22781	3290
67	0.7	10	251167074	20297	3283
68	0.7	10	703090523	18359	3269
69	0.7	10	733447773	19703	3276
70	0.7	10	399287411	19640	3229
71	0.7	100	1578708367	4797	3368
72	0.7	100	1001241461	312	3626

73	0.7	100	1119387339	1844	3506
74	0.7	100	814391786	1015	3585
75	0.7	100	152123184	1547	3596
76	0.7	100	912023484	594	3629
77	0.7	100	251167074	656	3546
78	0.7	100	703090523	1515	3567
79	0.7	100	733447773	2578	3539
80	0.7	100	399287411	2703	3460
81	0.7	500	1578708367	94	3578
82	0.7	500	1001241461	62	3574
83	0.7	500	1119387339	78	3625
84	0.7	500	814391786	141	3666
85	0.7	500	152123184	78	3656
86	0.7	500	912023484	78	3614
87	0.7	500	251167074	110	3682
88	0.7	500	703090523	109	3625
89	0.7	500	733447773	94	3658
90	0.7	500	399287411	93	3656

Time *: The 1000 is equal to 1 second.

After collecting above implementation results, it starts to do ANOVA. The work use Minitab to accomplish the job. There are three factors and we consider the interactions between pair-wise factors. The table 5.2 shows the ANOVA result of the QAP.

Table 5.2 The ANOVA for the QAP problem

Source	DF	Seq SS	Adj SS	Adj MS	F	P
<i>alpha</i>	2	238400	234598	117299	52.02	0
<i>finalTemperature</i>	2	2092597	2051253	1025627	454.82	0
<i>alpha*finalTemperature</i>	4	251497	245388	61347	27.2	0
<i>Random seed</i>	9	22628	24103	2678	1.19	0.332
<i>alpha*Random seed</i>	18	40350	39249	2180	0.97	0.514
<i>finalTemperature*Random seed</i>	18	56871	56871	3160	1.4	0.19
Error	36	81181	81181	2255		
Total	89	2783525				

The ANOVA tells us there are three places which cause significant difference, including the *alpha*, *finalTemperature*, and *alpha*finalTemperature*. We should notice that the pair-wise interaction between *alpha* and *finalTemperature* is significant difference, we should analyze it first rather than take them into consideration separately.

The study uses the following figure to present the interaction effect at figure 5.1. It shows when we fix the *finalTemperature* is 100, then there are significant difference when we apply the different cooling schedule. Otherwise, at the three treatments don't cause significant different under the *finalTemperature* is 10 and 500.

Finally, figure 5.2 shows when we apply the parameter *alpha* = 0.9 and *finalTemperature* = 10 or 100, it may be able to find better solution than others treatments.

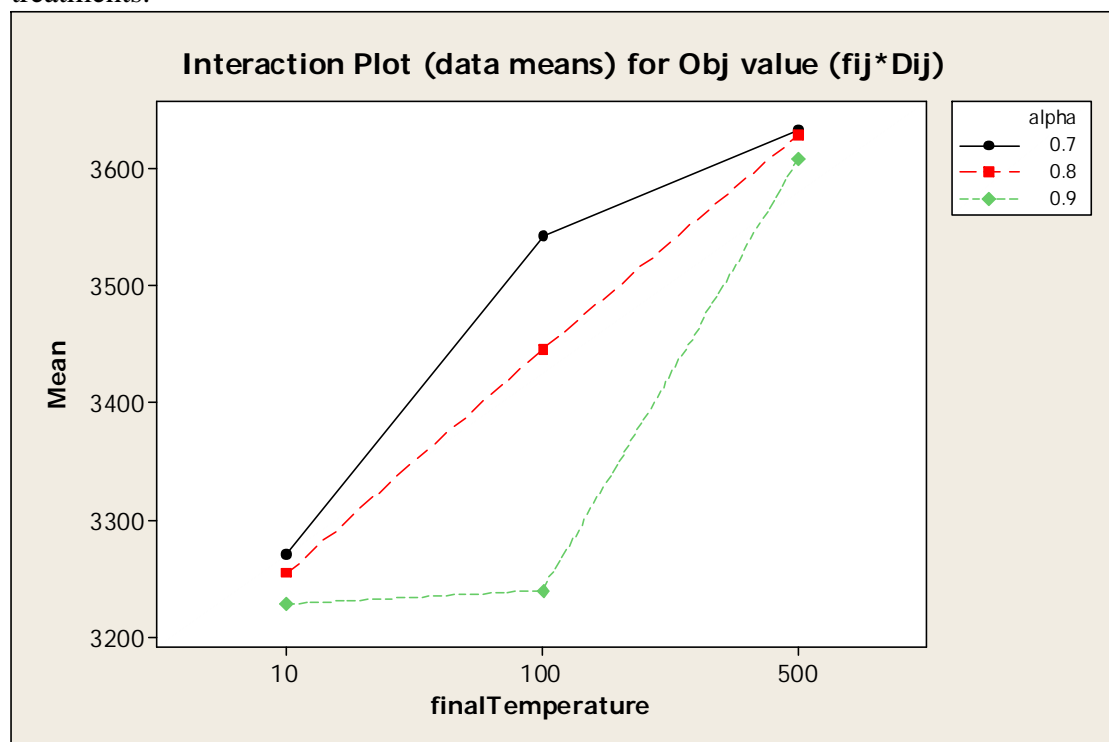


Figure 5.1 The interaction between *alpha* and *finalTemperature*

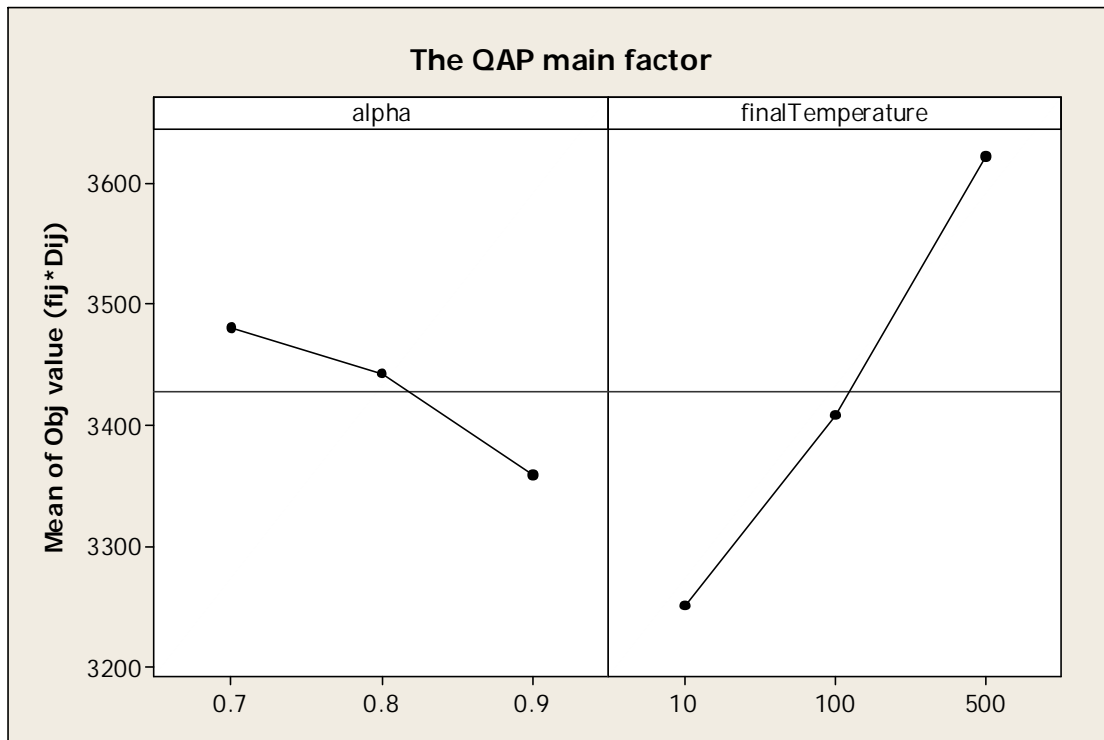


Figure 5.2 The main effect plot for the cooling schedule alpha and finalTemperature

5.2 The Experimental of Himmelblau Function and its ANOVA

The table 5.3 demons the experimental result of Himmelblau function and table 5.4 presents the ANOVA.

Table 5.3 the experimental result of Himmelblau function

Num	finalTemperat		Random seed	Time *	x_1	x_2	Obj value
	α	ure					
1	0.9	10	1578708367	15718	2.999	2.005	0.0003
2	0.9	10	1001241461	15250	2.995	2.006	0.0009
3	0.9	10	1119387339	15219	3.000	2.002	0.0001
4	0.9	10	814391786	15203	2.998	2.000	0.0001
5	0.9	10	152123184	15250	3.000	2.002	0.0001
6	0.9	10	912023484	15250	3.001	2.006	0.0007
7	0.9	10	251167074	15219	3.000	2.001	0.0000
8	0.9	10	703090523	15219	3.003	2.005	0.0009
9	0.9	10	733447773	15281	2.999	1.999	0.0001
10	0.9	10	399287411	15359	2.997	2.005	0.0004
11	0.9	100	1578708367	15297	2.999	2.003	0.0001
12	0.9	100	1001241461	15235	2.999	1.998	0.0002
13	0.9	100	1119387339	15250	2.996	2.007	0.0010
14	0.9	100	814391786	15218	2.998	2.005	0.0004
15	0.9	100	152123184	15250	3.000	2.011	0.0020
16	0.9	100	912023484	15344	3.000	2.000	0.0000

17	0.9	100	251167074	15281	3.001	2.001	0.0000
18	0.9	100	703090523	15219	3.002	2.003	0.0005
19	0.9	100	733447773	15281	2.996	1.999	0.0006
20	0.9	100	399287411	15266	3.001	1.998	0.0001
21	0.9	500	1578708367	31	3.012	1.981	0.0071
22	0.9	500	1001241461	16	2.864	2.088	0.5571
23	0.9	500	1119387339	15	2.974	2.012	0.0216
24	0.9	500	814391786	79	3.018	1.965	0.0205
25	0.9	500	152123184	15	2.915	1.962	0.3495
26	0.9	500	912023484	47	3.161	2.019	1.0861
27	0.9	500	251167074	31	2.937	2.150	0.3676
28	0.9	500	703090523	125	3.008	1.937	0.0589
29	0.9	500	733447773	32	2.938	2.042	0.1204
30	0.9	500	399287411	15	3.015	1.947	0.0392
31	0.8	10	1578708367	15266	3.001	1.999	0.0000
32	0.8	10	1001241461	15250	2.999	2.001	0.0001
33	0.8	10	1119387339	15266	3.002	2.002	0.0002
34	0.8	10	814391786	15250	3.001	2.002	0.0001
35	0.8	10	152123184	15266	2.998	1.997	0.0004
36	0.8	10	912023484	15297	3.000	2.000	0.0000
37	0.8	10	251167074	15312	3.003	1.996	0.0004
38	0.8	10	703090523	15234	3.000	2.000	0.0000
39	0.8	10	733447773	15297	3.005	1.988	0.0021
40	0.8	10	399287411	15235	3.000	2.000	0.0000
41	0.8	100	1578708367	11250	3.000	1.999	0.0000
42	0.8	100	1001241461	11547	3.003	2.005	0.0010
43	0.8	100	1119387339	9703	3.005	1.998	0.0007
44	0.8	100	814391786	10812	2.998	1.998	0.0003
45	0.8	100	152123184	10891	2.995	2.000	0.0009
46	0.8	100	912023484	10000	2.998	2.003	0.0002
47	0.8	100	251167074	7329	2.996	2.000	0.0006
48	0.8	100	703090523	10234	3.006	2.004	0.0023
49	0.8	100	733447773	10656	2.999	2.006	0.0005
50	0.8	100	399287411	10078	3.004	2.000	0.0006
51	0.8	500	1578708367	0	3.093	1.945	0.2809
52	0.8	500	1001241461	0	2.994	1.974	0.0161
53	0.8	500	1119387339	0	2.976	2.055	0.0475
54	0.8	500	814391786	31	3.030	1.956	0.0394
55	0.8	500	152123184	15	3.074	2.119	0.6453
56	0.8	500	912023484	32	3.063	1.978	0.1318
57	0.8	500	251167074	15	3.075	1.948	0.1788

58	0.8	500	703090523	32	2.949	1.826	0.7402
59	0.8	500	733447773	0	3.012	2.042	0.0474
60	0.8	500	399287411	62	3.008	1.988	0.0027
61	0.7	10	1578708367	15297	3.001	2.000	0.0000
62	0.7	10	1001241461	15281	3.002	1.992	0.0009
63	0.7	10	1119387339	15313	2.999	1.999	0.0001
64	0.7	10	814391786	15312	2.999	2.000	0.0000
65	0.7	10	152123184	15266	2.998	2.000	0.0002
66	0.7	10	912023484	15266	2.998	2.005	0.0003
67	0.7	10	251167074	15281	3.001	1.993	0.0008
68	0.7	10	703090523	15313	2.998	2.000	0.0001
69	0.7	10	733447773	15328	3.001	2.000	0.0000
70	0.7	10	399287411	15266	2.999	1.996	0.0003
71	0.7	100	1578708367	2796	3.000	2.004	0.0002
72	0.7	100	1001241461	2656	2.991	1.995	0.0044
73	0.7	100	1119387339	3016	2.999	1.984	0.0049
74	0.7	100	814391786	4562	3.001	1.999	0.0000
75	0.7	100	152123184	3844	3.013	1.989	0.0057
76	0.7	100	912023484	3969	3.004	1.992	0.0010
77	0.7	100	251167074	3469	2.994	2.003	0.0011
78	0.7	100	703090523	3968	3.003	2.006	0.0011
79	0.7	100	733447773	2422	2.996	2.002	0.0005
80	0.7	100	399287411	3547	3.004	1.986	0.0027
81	0.7	500	1578708367	15	3.116	1.905	0.4432
82	0.7	500	1001241461	0	3.628	-1.528	2.7571
83	0.7	500	1119387339	16	2.970	2.029	0.0307
84	0.7	500	814391786	0	2.949	2.217	0.7591
85	0.7	500	152123184	16	2.980	1.648	1.9367
86	0.7	500	912023484	0	2.964	2.134	0.2759
87	0.7	500	251167074	0	2.932	2.152	0.3821
88	0.7	500	703090523	0	3.110	1.807	0.6218
89	0.7	500	733447773	0	2.801	2.103	1.1634
90	0.7	500	399287411	0	3.090	1.674	1.2853

Time *: The 1000 is equal to 1 second.

Table 5.4 The ANOVA of the Himmelblau function

Source	DF	Seq SS	Adj SS	Adj MS	F	P
<i>alpha</i>	2	1.18544	1.18544	0.59272	6.14	0.005
<i>finalTemperature</i>	2	4.60276	4.60276	2.30138	23.85	0
<i>alpha*finalTemperature</i>	4	2.35656	2.35656	0.58914	6.1	0.001
<i>Random seed</i>	9	0.97801	0.97801	0.10867	1.13	0.37
<i>alpha*Random seed</i>	18	1.74195	1.74195	0.09677	1	0.479
<i>finalTemperature*Random seed</i>	18	1.94582	1.94582	0.1081	1.12	0.374
Error	36	3.47425	3.47425	0.09651		
Total	89	2783525				

By the ANOVA, we can understand the result is the same with QAP experiment, which *alpha*, *finalTemperature*, and *alpha*finalTemperature* cause significant difference. Therefore, we also do the analysis of *alpha*finalTemperature* and the figure 5.2 illustrates the interaction result. From the figure, we can see no matter under what kinds of cooling schedule, the treatments 10 and 100 of *finalTemperature* are overlap together which is also means there is no significant different. However, when it comes to the treatment 100 of *finalTemperature*, it is significant worse than others. Finally, the figure 5.4 shows the result of the two main factors which cause significant difference. Moreover, as for *alpha*, there is no difference between *alpha* = 0.8 or 0.9 although 0.8 is slightly better than 0.9. However, both of them are significantly better than 0.7. Then, the treatment, 10 and 100, of the *finalTemperature* that are the same and they are significantly better than the treatment of *finalTemperature* at 500.

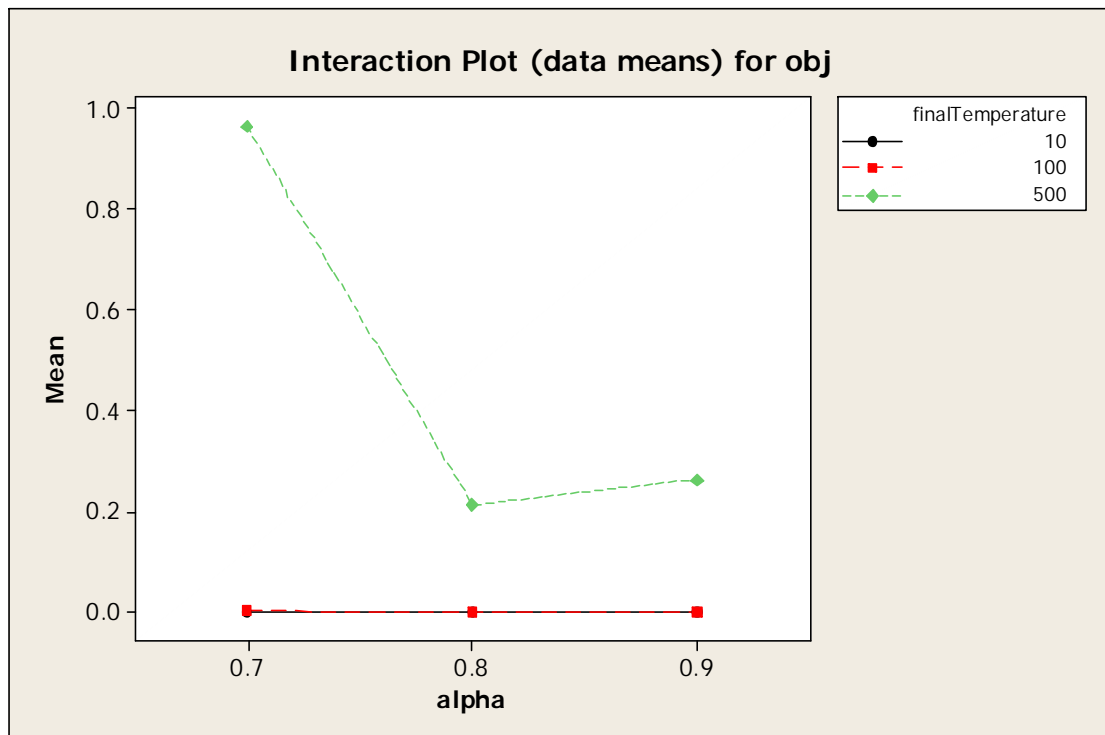


Figure 5.3 The interaction between the *alpha* and *finalTemperature*

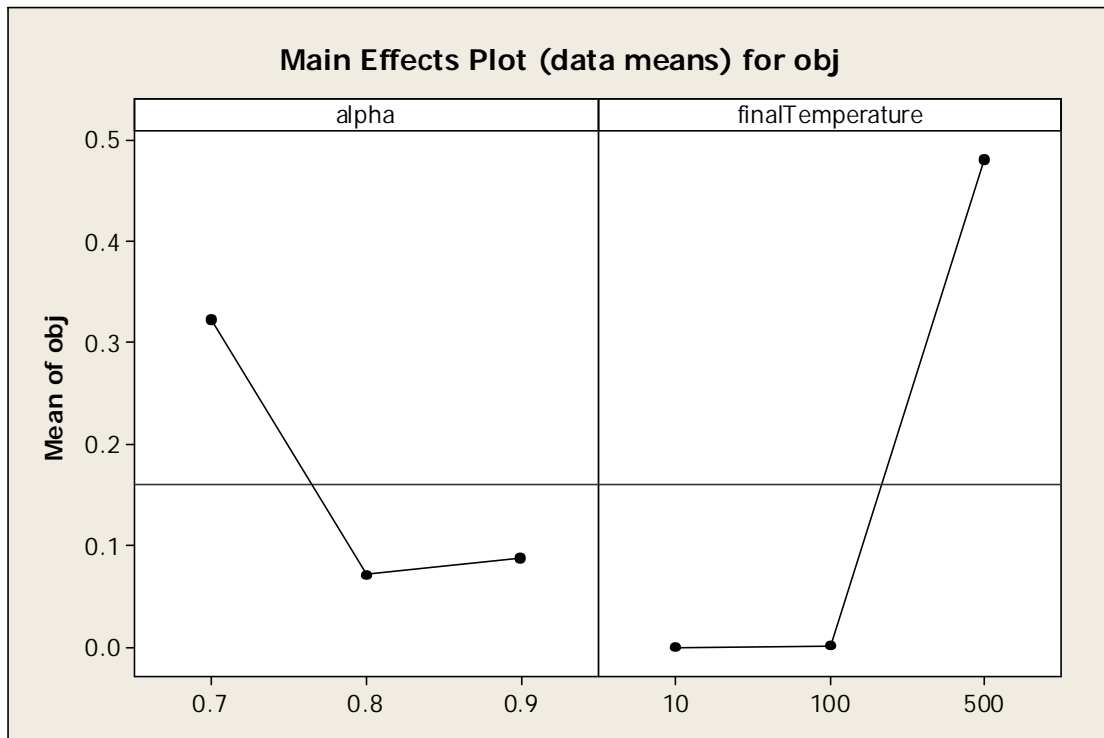


Figure 5.4 The main effect plot of the *alpha* and *finalTemperature*

6. Discussion and Conclusions

6.1 Discussion of the Study

After doing above experiment, we may find some places are worthwhile to be discussed, which are the factors that caused significant difference. First of all, the factors cause significant difference are the same at the combinatorial problem and the continuous problem. Moreover, there exists an interaction between the *alpha* and *finalTemperature*, it may mean we should consider it carefully. By the figure 5.1 and 5.3, it tells us when we choose higher *finalTemperature* (higher than 100), we have better not to select the cooling schedule which is also small (less than 0.8). Or it causes poor performance.

Second, the *finalTemperature* is smaller (less than 100) may lead to better solution. The reason is the algorithm that has more chance to explore solution space. Of course, too small *finalTemperature* is not very necessary because it may waste more computational time when it may has found better solution. Then, because the higher *finalTemperature* may stop too earlier when it does iteration, it is unable to yield good solution quality.

Finally, about the selection of the treatment of *alpha*, although the 0.8 at continuous problem may slightly better than 0.9, however, the treatment 0.9 is significantly better than 0.8. Therefore, the study still recommends solving different problems by 0.9. The table 6.1 is the suggested parameters for SA

Table 6.1 Suggested Parameters of SA

Factor	Treatment
Cooling schedule	0.9
Local search	At least 1000 for combinatorial problem
Initial temperature	1000
Final temperature	10 to 100

6.2 Conclusions

The work discusses the QAP and Himmelblau function, which are belonged to combinatorial problem and continuous problem respectively. We can understand some parameters are significantly influences the solution quality. Therefore, we can select them based on the findings from the experiment result.

There are some techniques are implemented here. First of all, we introduce the GA operator into the SA framework. For example, 1. Each solution is treated as a chromosome, 2. The local search operators are borrowed from the mutation operator of swap mutation, shift mutation, and the inverse mutation 3. Because the three local search operators modify the original solution into three different solution, it can be seen as a small population. Therefore, the study uses binary tournament to select better solution depending on their objective value. 4. The reheating technique is also applied in the work. Besides, we may restore the current solution back to current best when it encounters a number of iterations doesn't find better solution. Then, because both of them may also influence the solution quality, it can be as a future work to discuss them.

Finally, the study designs an object-oriented component, OpenSA, and integrated some techniques that is mentioned above to solve different problems. The interfaces of SA have been well-defined in the OpenSA. Besides, it may also be able to extend to solve the multiobjective problem. Therefore, it may provide a useful platform to do the research on the SA and solve variety problems.

Appendix: Nugent 30

```
int flow[][] = new int[][]
  {{0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7,3,4,5,6,7,8,4,5,6,7,8,9},
   {1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6,4,3,4,5,6,7,5,4,5,6,7,8},
   {2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5,5,4,3,4,5,6,6,5,4,5,6,7},
   {3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4,6,5,4,3,4,5,7,6,5,4,5,6},
   {4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3,7,6,5,4,3,4,8,7,6,5,4,5},
   {5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2,8,7,6,5,4,3,9,8,7,6,5,4},
   {1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7,3,4,5,6,7,8},
   {2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6,4,3,4,5,6,7},
   {3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5,5,4,3,4,5,6},
   {4,3,2,1,2,3,3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4,6,5,4,3,4,5},
   {5,4,3,2,1,2,4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3,7,6,5,4,3,4},
   {6,5,4,3,2,1,5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2,8,7,6,5,4,3},
   {2,3,4,5,6,7,1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7},
   {3,2,3,4,5,6,2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6},
   {4,3,2,3,4,5,3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5},
   {5,4,3,2,3,4,4,3,2,1,2,3,3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4},
   {6,5,4,3,2,3,5,4,3,2,1,2,4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3},
   {7,6,5,4,3,2,6,5,4,3,2,1,5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2},
   {3,4,5,6,7,8,2,3,4,5,6,7,1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6},
   {4,3,4,5,6,7,3,2,3,4,5,6,2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5},
   {5,4,3,4,5,6,4,3,2,3,4,5,3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4},
   {6,5,4,3,4,5,5,4,3,2,3,4,4,3,2,1,2,3,3,2,1,0,1,2,4,3,2,1,2,3},
   {7,6,5,4,3,4,6,5,4,3,2,3,5,4,3,2,1,2,4,3,2,1,0,1,5,4,3,2,1,2},
   {8,7,6,5,4,3,7,6,5,4,3,2,6,5,4,3,2,1,5,4,3,2,1,0,6,5,4,3,2,1},
   {4,5,6,7,8,9,3,4,5,6,7,8,2,3,4,5,6,7,1,2,3,4,5,6,0,1,2,3,4,5},
   {5,4,5,6,7,8,4,3,4,5,6,7,3,2,3,4,5,6,2,1,2,3,4,5,1,0,1,2,3,4},
   {6,5,4,5,6,7,5,4,3,4,5,6,4,3,2,3,4,5,3,2,1,2,3,4,2,1,0,1,2,3},
   {7,6,5,4,5,6,6,5,4,3,4,5,5,4,3,2,3,4,4,3,2,1,2,3,3,2,1,0,1,2},
   {8,7,6,5,4,5,7,6,5,4,3,4,6,5,4,3,2,3,5,4,3,2,1,2,4,3,2,1,0,1},
   {9,8,7,6,5,4,8,7,6,5,4,3,7,6,5,4,3,2,6,5,4,3,2,1,5,4,3,2,1,0}};

int distance[][] = new int[][]
  {{0,3,2,0,0,2,10,5,0,5,2,5,0,0,2,0,5,6,3,0,1,10,0,10,2,1,1,1,0,1},
   {3,0,4,0,10,4,0,0,2,2,1,0,5,0,0,0,2,0,1,6,1,0,1,2,2,5,1,10,5},
   {2,4,0,3,4,0,5,5,5,1,4,1,0,4,0,4,0,6,3,2,5,5,2,1,0,0,3,1,0,2},
   {0,0,3,0,0,0,0,2,2,0,6,0,2,5,2,5,1,1,1,1,2,2,4,0,2,0,2,2,5,5},
   {0,10,4,0,0,5,2,0,0,0,0,2,0,0,0,0,2,1,0,0,2,0,5,1,0,2,1,0,2,1},
   {2,4,0,0,5,0,1,2,2,1,4,10,10,2,5,5,0,5,0,0,0,10,0,0,0,4,0,10,1,1},
   {10,0,5,0,2,1,0,10,10,5,10,10,6,0,0,10,2,1,10,1,5,5,2,3,5,0,2,0,1,3},
   {5,0,5,2,0,2,10,0,1,3,5,0,0,0,2,4,5,2,10,6,0,5,5,2,5,0,5,5,0,2},
   {0,2,5,2,0,2,10,1,0,10,2,1,5,2,0,3,0,2,0,0,4,0,5,2,0,5,2,2,5,2},
   {5,2,1,0,0,1,5,3,10,0,5,5,6,0,1,5,5,0,5,2,3,5,0,5,2,10,10,1,5,2},
   {2,1,4,6,0,4,10,5,2,5,0,0,0,1,2,1,0,2,0,0,0,6,6,0,4,5,3,2,2,10},
   {5,0,1,0,2,10,10,0,1,5,0,0,5,5,2,0,0,0,0,2,0,4,5,10,1,0,0,0,0,1},
   {0,5,0,2,0,10,6,0,5,6,0,5,0,2,0,4,2,2,1,0,6,2,1,5,5,0,0,1,5,5},
   {0,0,4,5,0,2,0,0,2,0,1,5,2,0,2,1,0,5,3,10,0,0,4,2,0,0,4,2,5,5},
   {2,0,0,2,0,5,0,2,0,1,2,2,0,2,0,4,5,1,0,1,0,5,0,2,0,0,5,1,1,0}}
```

{0,0,4,5,0,5,10,4,3,5,1,0,4,1,4,0,0,3,0,2,2,0,2,0,5,0,5,2,5,10},
{5,0,0,1,2,0,2,5,0,5,0,0,2,0,5,0,0,2,2,0,0,0,6,5,3,5,0,0,5,1},
{6,2,6,1,1,5,1,2,2,0,2,0,2,5,1,3,2,0,5,1,2,10,10,4,0,0,5,0,0,0},
{3,0,3,1,0,0,10,10,0,5,0,0,1,3,0,0,2,5,0,0,5,5,1,0,5,2,1,2,10,10},
{0,1,2,1,0,0,1,6,0,2,0,2,0,10,1,2,0,1,0,0,5,2,1,3,1,5,6,5,5,3},
{1,6,5,2,2,0,5,0,4,3,0,0,6,0,0,2,0,2,5,5,0,4,0,1,0,0,0,5,0,0},
{10,1,5,2,0,10,5,5,0,5,6,4,2,0,5,0,0,10,5,2,4,0,5,0,4,4,5,0,2,5},
{0,0,2,4,5,0,2,5,5,0,6,5,1,4,0,2,6,10,1,1,0,5,0,0,4,4,1,0,2,2},
{10,1,1,0,1,0,3,2,2,5,0,10,5,2,2,0,5,4,0,3,1,0,0,0,5,5,0,1,0,0},
{2,2,0,2,0,0,5,5,0,2,4,1,5,0,0,5,3,0,5,1,0,4,4,5,0,1,0,10,1,0},
{1,2,0,0,2,4,0,0,5,10,5,0,0,0,0,0,5,0,2,5,0,4,4,5,1,0,0,0,0,0},
{1,5,3,2,1,0,2,5,2,10,3,0,0,4,5,5,0,5,1,6,0,5,1,0,0,0,0,0,0,10},
{1,1,1,2,0,10,0,5,2,1,2,0,1,2,1,2,0,0,2,5,5,0,0,1,10,0,0,0,2,2},
{0,10,0,5,2,1,1,0,5,5,2,0,5,5,1,5,5,0,10,5,0,2,2,0,1,0,0,2,0,2},
{1,5,2,5,1,1,3,2,2,2,10,1,5,5,0,10,1,0,10,3,0,5,2,0,0,0,10,2,2,0}};