Allocation Algorithm Performance

In case of sufficient optimal assets, the Algo will always find the best allocation within a short time (1 second);

In case of insufficient optimal assets, but sufficient available assets, the Algo will trigger the lpSolve solver and the run time depends on number of decision variables roughly around the number of margin calls times the number of available assets, and the constraints which determined by the amount of margin calls and assets, operational movements limit, the number of indivisible assets, and solver parameter settings.

**Tests**

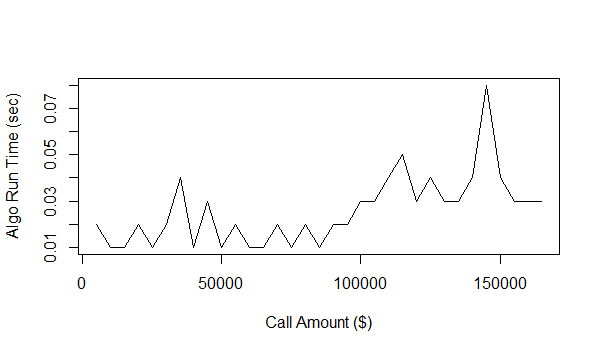
1. Call amount test

Input: 1 margin call, 3 assets

Independent variable: Margin call amount

Dependent variable: Algo run time

Result: There is a trend that running time increases as the call amount increases. When the call amount is small, the optimal assets are likely to be sufficient, the running time is shorter because the solver is not triggered. The average running time is around 0.02 seconds if solver is not triggered and 0.05 seconds with solver triggered.



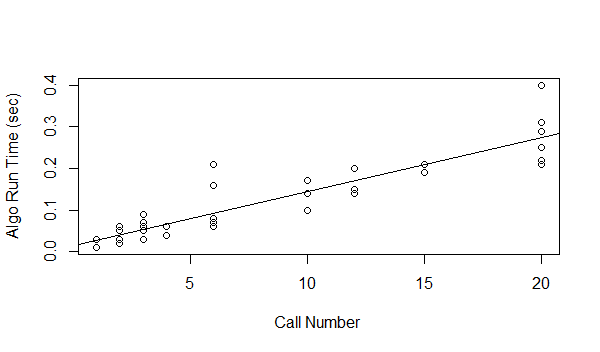
1. Call number test

Input: 3 assets per margin call

Dependent variable: number of margin calls

Independent variable: Algo run time

Result: The regression line shows an obvious trend that the running time increases with the increase of the number of margin calls.



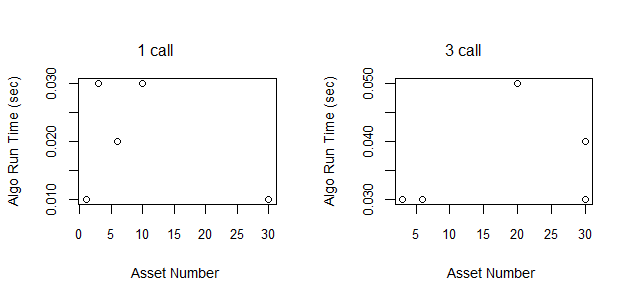
1. Asset number test

Input: fixed number of margin calls

Dependent variable: the number of asset

Independent variable: Algo run time

Result: The relationship between asset number and running time is not obvious when the number of margin calls is small. (To test: large number of calls)



1. Asset amount test

Input: 10 margin calls， 50 assets

Independent variable: 5 combinations the amount of the assets (small, medium, large, super large); 5 time limit settings (10s, 40s, 90s, 160s, 250s);

Dependent variable: Algo run time; objectives result; constraint result

Result:

1. Under 10 second time limit, the result can be found from the table below:

* Daily cost for all 5 assets combinations are the same;
* Reserved liquidity ratio are different because the total amount of the assets are different, so that fulfilling same amount of margin call will ends more or less assets in the inventory;
* Movements all meet the requirement (default setting is 2 per margin statement, not fungible)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Call Num | Asset Num Small | Asset Num Medium | Asset Num Large | Asset Num Super Large | Run Time | Daily Cost | RL Ratio | movement |
| 1 | 10 | 20 | 20 | 10 | 0 | 10.29 | 396.3164 | 0.8146 | 10 |
| 2 | 10 | 10 | 35 | 5 | 0 | 10.39 | 396.3164 | 0.6534 | 10 |
| 3 | 10 | 10 | 32 | 5 | 3 | 10.33 | 396.3164 | 0.8506 | 10 |
| 4 | 10 | 20 | 20 | 10 | 0 | 10.29 | 396.3164 | 0.8146 | 10 |
| 5 | 10 | 10 | 35 | 5 | 0 | 10.25 | 396.3164 | 0.6534 | 10 |

1. Under the same amount combination, the result can be found in the table below

* Daily cost, reserved liquidity ratio, and the movement are all the same under different time limit settings, which means it will take more than 250 seconds to find a better solution. That also indicates that, the first optimal solution will be found in a short time, so if we don’t have the time to let the solver run, we can just set the limit around 10 seconds.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Call Num | Asset Num Small | Asset Num Medium | Asset Num Large | Asset Num Super Large | Time Limit | Run Time | Daily Cost | RL Ratio | movement |
| 1 | 10 | 20 | 20 | 10 | 0 | 10 | 10.29 | 396.3164 | 0.8146 | 10 |
| 2 | 10 | 20 | 20 | 10 | 0 | 40 | 40.4 | 396.3164 | 0.8146 | 10 |
| 3 | 10 | 20 | 20 | 10 | 0 | 90 | 90.45 | 396.3164 | 0.8146 | 10 |
| 4 | 10 | 20 | 20 | 10 | 0 | 160 | 160.42 | 396.3164 | 0.8146 | 10 |
| 5 | 10 | 20 | 20 | 10 | 0 | 250 | 250.54 | 396.3164 | 0.8146 | 10 |

1. Movement test

Input: 10 margin calls, 50 fixed amount of assets

Independent variable: movements per margin statement and fungible setting

Dependent variable: Algo run time; constraint result

Result:

1. Under 10 second time limit, the result can be found from the table below:

* When 2 movement limit is set per margin statement, daily cost and reserved liquidity ratio are better with the fungible FALSE than TRUE. This result is interesting because there are more combinations when the fungible is true, but the result is worse. The reason is more combinations may lower the process of seeking the best solution.
* When fungible is false, increasing the movement limit per statement will improve the result in both cost and liquidity just a little bit, but it will take much more time. However, when fungible is false, in this case, the result are the same.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| movementPerMS | fungible | timeLimit | RunTime | dailyCost | RLRatio | movement |
| 2 | FALSE | 10 | 10.29 | 396.3164 | 0.814592 | 10 |
| 2 | TRUE | 10 | 10.36 | 402.5675 | 0.813071 | 10 |
| 4 | FALSE | 10 | 41.95 | 396.2995 | 0.814862 | 15 |
| 4 | TRUE | 10 | 32.31 | 402.5675 | 0.813071 | 10 |

1. When movement limit is fixed to 2. Compare the result in two tables below.

* When fungible is true, there is an improvement on the result at 160 second time limit, and a further improvement at 250 second. However, when the fungible is set to false, the result remains unchanged. This further confirmed that it takes more time for inputs that will generate more combinations.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| movementPerMS | fungible | timeLimit | RunTime | dailyCost | RLRatio | movement |
| 2 | TRUE | 10 | 10.36 | 402.5675 | 0.813071 | 10 |
| 2 | TRUE | 40 | 40.32 | 402.5675 | 0.813071 | 10 |
| 2 | TRUE | 90 | 90.39 | 402.5675 | 0.813071 | 10 |
| 2 | TRUE | 160 | 160.46 | 398.2953 | 0.813285 | 10 |
| 2 | TRUE | 250 | 250.31 | 398.2709 | 0.813288 | 10 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| movementPerMS | fungible | timeLimit | RunTime | dailyCost | RLRatio | movement |
| 2 | FALSE | 10 | 10.29 | 396.3164 | 0.814592 | 10 |
| 2 | FALSE | 40 | 40.28 | 396.3164 | 0.814592 | 10 |
| 2 | FALSE | 90 | 90.29 | 396.3164 | 0.814592 | 10 |
| 2 | FALSE | 160 | 160.3 | 396.3164 | 0.814592 | 10 |
| 2 | FALSE | 250 | 250.36 | 396.3164 | 0.814592 | 10 |

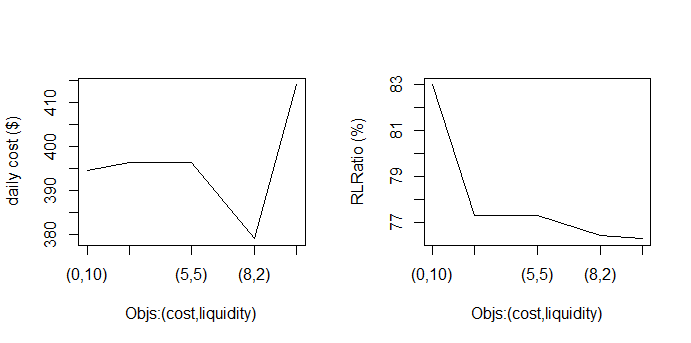
1. Objectives test

Input:

Independent variable: cost and liquidity settings

Dependent variable: Algo result accuracy (cost and liquidity)

Result: the cost is decreasing when the weight on cost is increasing; and the reserved liquidity ratio is increasing when the weight on liquidity is increasing.



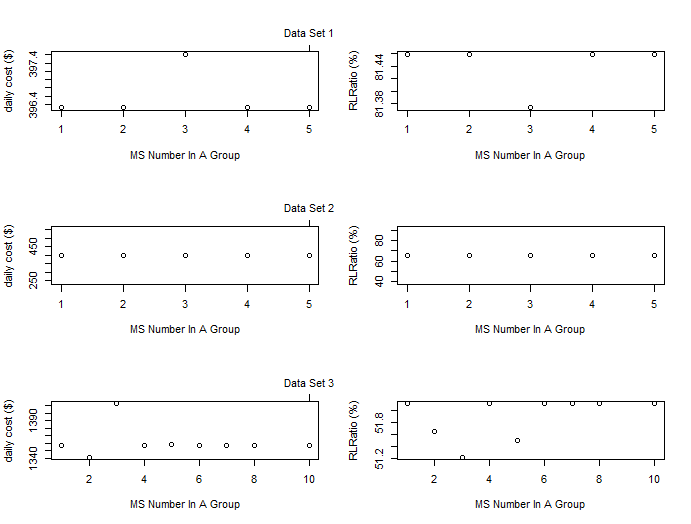
1. Number in group test

Input: 5 or 10 margin statements

Independent variable: number of margin statements in one group

Dependent variable: Algo result accuracy (cost and liquidity)

Result: not significant difference between different scenarios



1. Integral ratio change

Input: 10 margin calls and 50 assets

Independent variables: the ratio of indivisible assets, 3 combinations of 50 assets

Dependent variables: Algo run time and result accuracy (cost and liquidity)

Result: Running time has no difference below 250 seconds;

The figures are generated by the average of the result in 3 scenarios (assets amount combinations)

The figure on the left hand side shows us that the cost increases as the integral assets (indivisible assets) ratio increase. (No rounding done for the indivisible assets treated as divisible assets, further tests with the rounding are required)

While, the liquidity ratio doesn’t seems to have a linear relationship with the integral assets ratio, and the peak in the middle is probably due to the real situation.

