SOLVING A MULTI-DIMENSIONAL KNAPSACK PROBLEM USING A HYBRID PARTICLE SWARM OPTIMIZATION ALGORITHM

Nam Fai Wan
Lars Nolle
Nottingham Trent University
Clifton Lane, Nottingham, NG11 8NS
Email: nfwan5n@hotmail.com, lars.nolle@ntu.ac.uk

KEYWORDS

Particle Swarm Optimization, Multi-Dimensional Knapsack Problem

ABSTRACT

In this paper, an optimisation technique based on the Particle Swarm Optimization (PSO) algorithm will be experimented upon the Multi-dimensional Knapsack Problem. Through the merging of fundamental concepts of the existing PSO algorithm and selected features of evolutionary algorithms, a novel hybrid algorithm is created. When testing the algorithm against a test suite publicly available on OR-LIB, it was discovered that the algorithm is able to locate fitness values very close to best available results discovered using Linear Programming techniques, even though the algorithm is at the very early stage of development. Such an observation reveals the potential of this algorithm, calling for further research to be made upon it.

INTRODUCTION

The knapsack problem is an NP hard problem, and is part of a family of combinatorial problems (Martello and Toth, 1990). In the knapsack problem, the process of packing items with cost and profit associated to them into a container is simulated. Typically, the aim is to maximise the fitness, at the same time not exceeding supplied constraints. Given its simplicity, it can be used to test the performance of an algorithm in a discreet search space, and can act as a model for practical applications such as financial portfolio optimization and engineering design optimization.

Amidst the many different types of knapsack problems is the Multi-dimensional Knapsack Problem (MKP), which extends upon the basic knapsack problem with an additional dimension (Chu and Beasley, 1998). Instead of having just one constraint, the number of constraints can increase, which makes it possible to simulate a larger range of real-world optimization problem. The MKP can be described mathematically as follow:

maximise

$$f = \sum_{i=1}^{n} f_i \cdot \delta_i \tag{1}$$

given equations

$$B_j \geq \sum_{i=1}^{n} u_{ij} \cdot \delta_i \tag{2}$$

$$\delta_i \in \{0,1\}, \quad i = 1, ...n, \quad j = 1, ...m \tag{3}$$

Legend
i = identifier of candidate items
n = number of candidate items
j = identifier of constraints
m = number of constraints
f = fitness value
u_{ij} = cost value of item i, constraint j
B_j = maximum cost value of constraint j
δ_i = 0/1 value determining if the item i is in the knapsack

LITERATURE SURVEY

In this section, a detailed review on existing literature will be presented in a systematic manner. Existing attempts that tackle the MKP is initially presented, followed by fundamental concepts of the original Particle Swarm Optimization (PSO) algorithm.

Previous Attempts on the MKP

A quick survey on existing literature reveals that a range of well known stochastic algorithms have been tested on the MKP. In their paper, Chu and Beasley (1998) attempted to solve the MKP using Genetic Algorithms, and obtained good results using less computational effort as opposed to many other techniques which include the CPLEX MIP solver. While the results are not as good as those obtained using Linear Programming techniques, which serve as the benchmark in their test, it demonstrates that a stochastic algorithm is able to locate results that are very close to the known optimum under less time, and is useful for applications where the efficiency of an algorithm is a decisive factor. This success led other researchers to test difference stochastic algorithms

Proceedings 23rd European Conference on Modelling and Simulation ©ECMS Javier Otamendi, Andrzej Bargiela, José Luis Montes, Luis Miguel Doncel Pedrera (Editors)
ISBN: 978-0-9553018-8-9 / ISBN: 978-0-9553018-9-6 (CD)

against the MKP, for example, Qian and Ding (2007), who tested the Simulated Annealing algorithm against the Chu and Beasley test suite and Hembecker et.al. (2007), who tested the PSO algorithm against smaller instances of the same test suite. While it was reported in both work that Simulated Annealing and PSO are unable to locate results as good as the one located using Genetic Algorithm, the test results are still valuable as it give researchers an indication of heuristics that are less suitable in solving the MKP, and provide inspiration to create a hybrid algorithm that contains the strengths of different algorithms.

The Original PSO Algorithm

PSO is fundamentally a stochastic, population-based search algorithm which mimics organisms that interact as a swarm such as a school of fish or a swarm of bees looking for food. Its creators (Kennedy and Eberhart, 1995) found elegance in design by observing the social interaction and learning behaviour of swarm-based animals; the animals do not travel randomly in search for food but have an observable pattern, where information such as the nearest 'patch' of food is shared among individuals in the swarm.

With the above heuristic in mind, Kennedy and Eberhart later created a mathematical formula to mimic the behaviour above, as seen in notations (4) and (5).

$$\begin{aligned} v_{a,b} \leftarrow\ & c_1 r_1 (gbest_b - x_{a,b}) \\ & + c_2 r_2 (pbest_{a,b} - x_{a,b}) \end{aligned} \tag{4}$$

$$x_{a,b} \leftarrow x_{a,b} + v_{a,b} \tag{5}$$

Legend
$v_{a,b}$ = Velocity of *a*th Particle in *b*th Dimension
$x_{a,b}$ = Position of *a*th Particle in *b*th Dimension
c_1, c_2 = Constant parameters controlling influence
r_1 ,r_2 = Random number
$gbest_b$ = Position of global best particle
$pbest_{a,b}$ = Position of particle's previous best position

To implement the PSO algorithm in a computing environment, the algorithm's particles, or agents, are encoded to represent candidate solutions in an n-dimension search space. Each agent has an associated velocity, which dictates how the agent travels in the search space. By calculating the position of an agent based on its velocity, new candidate solutions in each iteration or measured timeframe can be obtained. Also, each agent will have knowledge of its previous best fitness and the global best fitness of the swarm. This knowledge will be used to influence the velocity of the agents, and can be seen in mathematical notation (4). With constant parameters that control global influence and local influence, it allows the user to have the flexibility in controlling the algorithm's behaviour. On the other hand, random numbers reduce the chances of premature convergence by introducing the element of randomness. Using the above heuristic, the velocity of the agents will be calculated, and the agents will move according to a certain velocity using the formula illustrated in notation (5). This will be repeated until a stopping condition is met.

By studying the above algorithm, it can be seen that the original PSO is intended for problems which are contiguous in nature. In 1997, Kennedy and Eberhart proposed a customised version of their PSO algorithm to tackle problems in a discrete search space. In this version, the trajectories of agents are calculated from the probability of whether the next coordinate will be either one or zero. Instead of using notation (5) to update the velocity and position of the particles, the sigmoid function in (6) is used, and (7) calculates the trajectory in the binary search space.

$$s(v_{a,b}) = \frac{1}{(1 + \exp(-v_{a,b}))} \tag{6}$$

$$\text{if } r < s(v_{a,b}) \text{ then } x_{a,b} = 1;\ \text{else } x_{a,b} = 0 \tag{7}$$

Legend
r = Random number

While the original discrete PSO is successful in many cases, it is interesting to see if it can be simplified by replacing some of its operators with those found in evolutionary algorithms while still maintaining the behaviour demonstrated by PSO. One of the reasons why this is being looked into is because the probability of a bit being 1 or 0 is calculated and later assigned to every bit in all agents of the swarm, and this operation may be computationally expensive. Wang (2007) approached this problem by replacing (6) and (7) with operations based on Estimation of Distribution - a technique which generates new solutions using probabilistic models of the distribution of regions, and have shown that in their test cases is superior compared to the algorithm proposed by Kennedy and Eberhart (1997). In the paper, the simplification process will be taken further, where a Hybrid PSO algorithm that is partly inspired by the use of inheritance operators in Tseng and Liao (2008) will be proposed in the next section.

PROPOSED IDEAS

In this section, a number of new customisations of the PSO algorithm are proposed. By using these new and novel concepts, search algorithms that have the strengths of different algorithms in this area of research are created.

A Hybrid PSO Algorithm

In this paper, an operator in evolutionary algorithm called inheritance, which is illustrated in Figure 1, will be adapted for the PSO algorithm to replace the position updating operation based on velocity and probability. Such a technique relies on the idea that the more similar two agents are, the closer they are to each other.

Essentially, it is proposed that a controlled number of bits are selected randomly from the parent agent, and applied to the corresponding position of the child agent. This mechanism can be applied onto the heuristics where information from the global best agent (gbest) and information from previous memory (pbest) needs to be transferred to a particular agent, which can be controlled by the bits to copy from gbest and bits to copy from pbest parameters respectively. This is possible as the one-way sharing mechanism allows information of a child to be inherited from multiple sources, in which a child can have multiple parents, making it not just limited to the parents mentioned above. Furthermore, with parameters that control the number of bits to inherit, it gives the user the ability to control the behaviour of the algorithm, thus allowing the flexibility of building highly dynamic and complex algorithms that rely on a large number of heuristics. Do note that as the parameter can be changed during runtime, it is possible to dynamically change the behaviour of an algorithm when it is executing, though this significantly complicates the parameter tuning process.

Besides the inheritance operator, it is also proposed that an adapted mutation operator is used to introduce the element of randomness into the algorithm, substituting the r_1 and r_2 variables in the formula proposed in notation (4). This mutation operator works by random flipping a bit of the search agent, where the chance of flipping is determined by a parameter called mutation rate. One of the key benefits of the mutation operator proposed as opposed to the random variables r_1 and r_2 of the original PSO is the degree of randomness that can be introduced is controllable in a separate manner. In order to encourage exploration, a high mutation rate value can be set. Setting a low value reduces the stochastic factor, and allows other heuristics to affect the algorithm to a higher extent. It needs to be noted that this is different from the v_{max} parameter used by Kennedy and Eberhart (1997), as the v_{max} parameter controls the distance an agent can ultimately travel, whereas the proposed parameter controlling the degree of mutation does not directly affect the inheritance operation.

Repair Operation

By using the algorithm above, it is possible that an agent can contain a solution that does not conform to its constraints. In order to overcome this problem, a random bit that contains the value 1 is switch to the value 0, and a fitness evaluation is then performed to ensure the value is within its constraints. This is then repeated until the solution satisfies all constraints, and the entire operation resembles the action of taking out items from the knapsack until the no constraints are violated. As this operation is considered a significant part in locating a solution, it is decided that every repair operation performed is measured and included in the results related to the efficiency aspect of the algorithm.

EXPERIMENT FRAMEWORK

The purpose of this section is to document the steps involved in the implementation and testing of the ideas proposed in previous chapters. Reproducibility of the experiments is of primary concern due to the stochastic nature of the algorithms tested.

The OR-LIB Test Suite

For comparison and analysis purposes, the MKP test suite published on OR-LIB by Chu and Beasley (1998) is being used for testing the performance of the algorithm proposed earlier. There are a number of reasons why this approach was chosen. Firstly, using a publicly accessible test suite makes it possible for the reader to compare presented results against other results published in other literature. Doing so allows the strengths and weaknesses of the proposed algorithm to be identified, facilitating the decision making process when choosing a suitable algorithm. Secondly, such an approach also facilitates the reproduction of the experiment compared to approaches where the test data is generated dynamically. One of the concerns of using dynamically generated test data is that the tested algorithm may behave differently, due to the difference in test data. Having a consistent benchmark removes this concern, allowing the tested algorithms to be compared more fairly.

Measurement Approach

In order to evaluate the effectiveness of the algorithm, the best known fitness values, which are obtained using Linear Programming (LP) techniques and included in the test suite, are used as a benchmark. Using the formula in (8), the gap between the best known results and the results presented can be obtained. As the derived value is a percentage value, having a smaller value indicates that the obtained result is closer to the best known result, in which having a gap value of 0% means the algorithm succeed in obtaining the best known result.

$$\frac{f_{bestknown} - f_{obtained}}{f_{bestknown}} \times 100 \tag{8}$$

With regard to the efficiency aspect of the algorithm, the number of total fitness evaluations performed in order to achieve the best obtainable value is used as a measurement. The diligent reader might notice that this approach is different compared to the approach used in other literature, where efficiency is usually measured in terms of execution time. There are a number of reasons why this approach is not chosen. Firstly, the environment in which

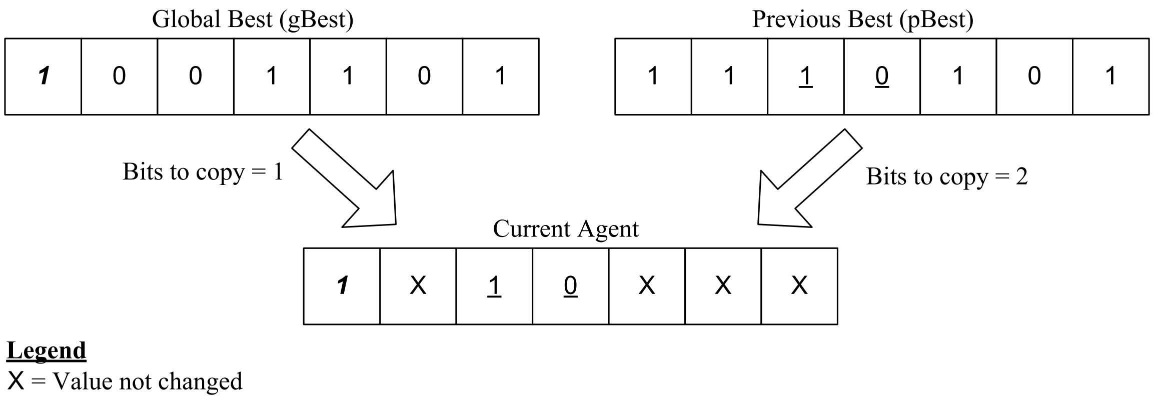

Figure 1: Parameter Controlled Position-Point Inheritance

the experiment is run on can significantly affect its execution time. There are differences in hardware such as CPU speed, CPU types and differences in software such as the Operating System used, influence of other programs in a multi-tasking systems, and programming platform that can skew the result. Because it is extremely difficult to recreate an environment that is exactly the same as the one it is run on, a measurement that is not affected by these factors is selected. While it can be argued that doing so makes it difficult to compare the efficiency of the algorithm against results published in existing literature, this can be overcame by recreating and testing candidate algorithms in the same environment.

Algorithm Parameters

Being a novel algorithm, parameters for the proposed algorithm need to be identified, and this is one of the challenges in getting the algorithm to work in an optimum manner. For this experiment, the parameters used are derived from the author's previous work, where the proposed algorithm was tested on a much smaller knapsack and a set of parameters is obtained using a simple parameter tuning process (Wan, 2008). Due to differences in the size of the knapsack (n), the parameters are scaled according to the size of the knapsack, and are listed in Table 1. With regard to the stopping condition, the experiments in this paper terminate once 15×10^6 fitness evaluations are made, which include fitness evaluations made during repair operations. While it can be argued that such a parameter tuning process does not guarantee the best set of parameters is obtained, it serves as a good starting point for setting up initial experiments. Once the proposed algorithm has been better understood, work should be conducted in this area to ensure that a set of parameters is readily available.

RESULTS AND DISCUSSION

The experiment results are presented in Table 2, and this is accompanied by test results obtained from work by Chu and Beasley (1998) and Qian and Ding (2007) for comparison purposes. By comparing the average of the average % gap values for each algorithm located at the bottom of the table, it can be seen that the proposed algorithm is able to locate results that are better than results from Qian and Ding, but is slightly less favourable as opposed to results from Chu and Beasley. While this may indicate that the Genetic Algorithm implemented by Chu and Beasley is able to perform better than the proposed algorithm for the MKP, the reader needs to bear in mind that this is an algorithm still in the early stage of development, and there is room to further improve the proposed algorithm. Firstly, it needs to be stressed that the parameter tuning process used is very simplistic, and it is possible that using a set of parameters obtained from a better parameter tuning process can yield better results. Secondly, the repair operator used by the proposed algorithm is based on a very simple heuristic, whereas the repair operator used in Chu and Beasley is complex and expensive in computational term, for example, the number of fitness evaluations needed. It is possible that the proposed algorithm can be improved by replacing the repair operator with the one proposed by Chu and Beasley. Finally, due to the simplicity and extensibility of the proposed algorithm, it is possible to invent new heuristics and append it to the proposed algorithm. A list such of heuristics exists, and is documented in previous work (Wan, 2008).

Table 1: Parameters for Proposed Algorithm

Name	Value
Bits to copy from gbest	8% of n
Bits to copy from pbest	30% of n
Mutation Rate	0.33
Swarm Size	150% of n

Key findings aside, analysis of the results obtained reveals that similar to results obtained by Chu and Beasley and Qian and Ding, the proposed algorithm has the highest average percentage gap for problems with small knapsack size, high number of constraints and small tightness ratio ($m = 30$, $n = 100$, $\alpha = 0.25$), indicating that this a set of test instances that are difficult to solve. With regard to the efficiency aspect of the algorithms, it is unfortunate that different approaches are used to measure the efficiency of a search, hence making a fair comparison between different algorithms very difficult. Nevertheless, it can be seen that for the proposed algorithm, the number of fitness evaluations required to locate the best solution increases as the size of the knapsack grows larger, and it is reasonable to believe that this applies to knapsacks with size larger than 500. Because the efficiency of an algorithm still plays a significant role when deciding if it is suitable for an application, future work that evaluates the efficiency of a range of algorithm is need, with great emphasis given to maintain a consistent execution environment that is able to provide accurate measurements.

CONCLUSION

Based on the key findings, it can be concluded that the proposed algorithm is able to locate a fitness value that is close to the best fitness values reported in the literature, which were obtained using linear programming. Seeing that this simple, extensible algorithm is still in the early stage of development, there exist the potential to increase its performance with better sets of parameters or addition of new heuristics. It is hoped that these finding will inspire future researchers in analysing the proposed algorithm from different perspectives to further improve upon it.

REFERENCES

Chu, P. C., Beasley, J. E., 1998. A Genetic Algorithm for the Multidimensional Knapsack Problem, *Journal of Heuristics*, 4(1), 63-86.

Hembecker, F., Lopes, H. S., Godoy, W., 2007. Particle Swarm Optimization for the Multidimensional Knapsack Problem, *In: Belczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B., ed. Proceedings of the 8th international Conference on Adaptive and Natural Computing Algorithms, Part I, Warsaw, 11 April-14 April 2007*, Berlin, Heidelberg:Springer-Verlag, pp.358-365.

Kennedy, J., Eberhart, R. C., 1995. Particle Swarm Optimization. *In: IEEE, ed. Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, 27 November-1 December, 1995*, Piscataway: IEEE service center, pp.1942-1948.

Kennedy, J., Eberhart, R. C., 1997. A discrete binary version of the particle swarm algorithm. *In: IEEE, ed. IEEE International Conference on Computation Cybernetics and Simulation, Orlando, 12-15 October, 1997*, Piscataway: IEEE service center, pp.4104-4108.

Martello, S., Toth, P., 1990. *Knapsack Problems: Algorithms and Computer Implementations*. Chichester:John Wiley and Sons.

Tseng, C., Liao, C., 2008. A discrete particle swarm optimization for lot-streaming flowshop scheduling problem, *Computers and Operations Research*, 191(2), 360-373.

Qian, F., Ding, R., 2007. Simulated Annealing for the 0/1 Multidimensional Knapsack Problem, *Numerical Mathematics*,16(4), 320-327.

Wan, N. F., 2008. *The Particle Swarm Optimisation Algorithm and the 0-1 Knapsack Problem*. MSc thesis, Nottingham Trent University.

Wang, J., 2007. A Novel Discrete Particle Swarm Optimization Based on Estimation of Distribution, *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, 2486, 791-802.

AUTHOR BIOGRAPHIES

NAM FAI WAN was born in Kuala Lumpur, Malaysia and went to Nottingham Trent University, where he studied software engineering and obtained his BSc degree in 2004. He worked for a software house in Malaysia that produces financial solutions until 2006, and then returned to Nottingham Trent University to take up a Masters degree in Internet and enterprise computing, which he completed in 2008. His email is `nfwan5n@hotmail.com`.

LARS NOLLE is a Senior Lecturer at the Nottingham Trent University. He is Programme Leader and Head of the Computational Optimisation and Applications Research Group. His research interests include computational intelligence for scientific and engineering applications, computer security and software engineering. His email is `lars.nolle@ntu.ac.uk`.

Table 2: Computational Results from Chu and Beasley (1998), Proposed Algorithm and Qian and Ding (2007)

Problem			Chu and Beasley (1998)	Proposed Algorithm		Qian and Ding (2007)
m	n	α	Average % Gap	Average % Gap	AFEBS	Average % Gap
5	100	0.25	0.99	1.16	4,431,497	2.95
		0.50	0.45	0.53	3,848,719	1.67
		0.75	0.32	0.36	3,792,008	0.87
5	250	0.25	0.23	0.58	8,609,279	3.09
		0.50	0.12	0.25	10,281,679	1.89
		0.75	0.08	0.15	7,334,528	0.84
5	500	0.25	0.09	0.89	10,169,920	3.41
		0.50	0.04	0.33	10,306,961	2.04
		0.75	0.03	0.17	10,827,946	0.92
10	100	0.25	1.56	2.22	7,351,304	3.93
		0.50	0.79	1.01	4,615,686	2.09
		0.75	0.48	0.61	3,934,004	1.06
10	250	0.25	0.51	1.09	9,693,256	3.21
		0.50	0.25	0.47	9,203,460	2.14
		0.75	0.15	0.27	8,024,108	1.01
10	500	0.25	0.24	1.30	10,302,268	3.67
		0.50	0.11	0.56	11,252,190	2.50
		0.75	0.07	0.26	11,243,238	1.17
30	100	0.25	2.91	3.69	3,263,436	4.94
		0.50	1.34	1.62	4,621,375	2.62
		0.75	0.83	1.04	1,993,596	1.29
30	250	0.25	1.19	1.91	7,331,979	3.60
		0.50	0.53	0.77	8,692,707	2.25
		0.75	0.31	0.46	7,013,305	0.96
30	500	0.25	0.61	1.83	10,434,576	3.75
		0.50	0.26	0.75	10,706,743	2.30
		0.75	0.17	0.37	10,489,562	1.07
		Average	0.54	0.91	7,769,234	2.27

AFEBS = Average number of Fitness Evaluations to obtain Best Solution