Solving Knapsack Problems using Radius Particle Swarm Optimization fuse with Simulated Annealing

Mudarmeen Munlin
Faculty of Engineering and Technology
Mahanakorn University of Technology
Bangkok, Thailand
mmunlin @gmail.com

ABSTRACT[1]

We present a novel approach to fuse the Radius Particle Swarm Optimization and Simulated Annealing (RPSO-SA) to solve the Knapsack Problems (KPs). The features RPSO-SA create an innovative approach, which can generate high-quality solutions in shorter times and more stable convergence characteristics. The RPSO takes advantage of group-swarm to keep the balance between the global exploration and the local exploitation. The SA gently improves the candidate solution by searching for optimal solutions within a local neighbourhood. The RPSO-SA combines the strong global search ability of RPSO and the strong local search ability of SA to reach faster optimal solution. In addition, there are two ways of accepting a new solution. The method has been tested against the knapsack problems. The results indicate that the combined approach outperforms individual implementations of radius particle swarm optimization and simulated annealing.

Keywords—Radius Particle Swarm Optimzation; Simulated Annelling; Combined Algorithm; Knapsack Problem.

1. INTRODUCTION

The Knapsack Problems)KPs (are NP-hard that has been studied in the last few decades, involving both and exact algorithms and approximated methods. The exact approaches include branch and bound [1] and dynamic programming [2]. The approximated mainly include the hybrid algorithm [3] and meta-heuristic algorithms such as radius particle swarm optimization (RPSO) [4], simulated annealing (SA) [5] and genetic algorithm (GA) [6].

KPs are optimization problems in the binary domain and are used for variety of practical problems, such as capital budget control, resources allocation [7], project selection [8], cutting and packing problems [9] and cryptography [10]. The KP is given a set of items with weights and sizes and the capacity value of a knapsack in order to maximise the total weight of selected items in the knapsack to satisfy the capacity constraint. Due to its practical values, Thiongane et al. [11] show the combination method to deal with lagrangian heuristics, local searches, variable fixing and re-optimization by a Sub-Gradient Algorithm (SGA). Bansal and Deep [12] propose the Modified Binary PSO (MBPSO) to update the MBPSO position term and to provide a new probability of selection. Zhang et al. [13] converted the KPs to a directed graph using the Network Converting Algorithm (NCA). Wang et al. [14] propose an effective hybrid algorithm based on Estimation of Distribution Algorithms (EDAs) that included a new probability model based on specific knowledge to improve the convergence speed.

[1] I wish to thank Dr.Samart Moodleah for invaluable comments and final proofreading of this paper.

We present a novel approach that combined the radius particle swarm optimization and simulated annealing (RPSO-SA) for solving the KPs. The results of the proposed method will compare with the individual radius particle swarm optimization (RPSO) and simulated annealing (SA).

2. RELATED WORK

This section introduces the knapsack problem (KP), the radius particle swarm optimization (RPSO) and the simulated annealing algorithm (SA) as well as their equations.

Knapsack Problem

There has been a lot generalization of the classical knapsack problem, depending upon the distribution of the items. In this paper, we focus the KPs such that 0-1 knapsack problem (KP): each item may be chosen at most once and multi-dimension knapsack problem (MKP): If we have *n* items and *m* knapsacks with capacities not necessarily same and knapsack are to be filled simultaneously.

0-1 Knapsack: In the classical knapsack problem (KP) we have a set of items ($i = 1, \dots, n$) and a knapsack of limited capacity. Each item we associated a positive profit p_i and a positive weight w_i. The problem for finding the set of items with maximum overall profit among those whose total weight does not exceed the knapsack capacity C. The problem may be formulated so as the maximize the total profit f(x) as follows;

$$Maximize\ f(x) \sum_{i \in n} p_i x_i \qquad (1)$$

$$Subject\ to\ \sum_{i \in n} w_i x_i \leq c\ ;\ x_i \in [0,1], i = 1, \dots n$$

Where x_i is a binary number, is equal 1 if and only if item i is selected. However, we cannot take all items because the total weight more than the knapsack capacity.

Multi-dimension Knapsack: The multi-dimension knapsack problem (MKP) is a generalization of the classic knapsack problem. It consists of selecting a subset of given items in such a way that the total profit of the selected items is maximized. Therefore, the problem can be given follows;

$$Maximize\ f(x) \sum_{i \in n} p_i x_i \qquad (2)$$

$$Subject\ to\ \sum_{i \in n} w_{i,j} x_i \leq C_j \quad \forall j = 1, .., m$$

$$x_i \in [0,1], i = 1, \dots n$$

Where n is the number of items, m is the number of knapsack constraints with capacities C_j, p_i is the positive profit, x_i is a binary number, is equal 1 if and only if item i is selected and $w_{i,j}$ is the positive weight of the knapsack's constraints matrix.

Radius Particle Swarm Optimzation Algorithm (RPSO)

Radius Particle Swarm Optimization (RPSO) [4] is proposed by extending the ring topology to include a set of neighbors using a radius-based neighborhood in order to speed up convergence and avoid the local optima. The RPSO is a simple concept, easy to implement and computationally inexpensive. It extends the traditional PSO algorithm by grouping particles within the same radius into a new particle agent and then iteratively finding the best solution under the given objective functions. The significant concept of the RPSO is based on finding the agent particle within the radius of a circle. Thus, it uses a swarm circle topology to find the agent particle within the radius of the circle. Each particle in the overlap radius can be in multiple groups. Once a group is defined, it finds the best particle in that swarm group and assign to the agent particle. Finally, the agent particles are the candidates for finding the optimal solution, or the *gbest* position. To overcome the premature convergence problem, the RPSO takes advantage of group-swarms to maintain the swarm diversity and evolution by sharing information from the agent particles, which effectively keep the balance between the global exploration and the local exploitation. Obviously, the agent particle guides the neighbouring particles to jump out of the local optimum and achieve the global best.

The Euclidean distance is used to calculate the radius-neighbourhood between particle i and particle θ using equation 3.

$$d(p_i, p_{\emptyset}) = \sqrt{\sum_{j=1}^{m} \left(p_{i,j} - p_{\emptyset,j}\right)^2} \; ; d \leq 2r \, , i \neq \emptyset \;\; , for \;\; 1 \leq i \leq n \;\; , for \;\; 1 \leq \emptyset \leq n \tag{3}$$

where

$$p = \{x_1, x_2, x_3, \dots, x_n\}$$

$$r = \mu \cdot v_{max}; \mu[0.0,1.0] \tag{4}$$

In equation (4), the radius value (r) of the particle is obtained. Here, r is determined by the maximum velocity (v_{max}). Therefore, v_{max} is assigned to the maximum bounds of the search space or the feasible bounds in the benchmark function. We consider the problem of finding the global optimum using the agent particle $(\text{abest}_{i,j})$ within a radius-neighbourhood as given in equation (5).

$$abest_{i,j} = \min_{\beta \in \rho} f(\beta) \tag{5}$$

Therefore, the particle i in the swarm updates it velocity and position as given in equation (6) and equation (7), respectively.

$$v_{i,j}(t+1) = w \cdot v_{i,j}(t) + c_1 \cdot R_1 \cdot \left(pbest_{i,j} - x_{i,j}(t)\right) + c_2 \cdot R_2 \cdot \left(abest_{i,j}(t) - x_{i,j}(t)\right) \tag{6}$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \tag{7}$$

Simulated Annealing Algorithm (SA)

Simulated annealing (SA) is a gradient method for the global optimization problem which is firstly proposed by Kirkpatrick et al. [5]. This algorithm gradually improves the solution by searching for better solutions within a local neighborhood. There are two ways of accepting a new solution: (1) if its fitness value is better than that of the current solution; (2) by accepting a solution with a worse fitness value with a certain probability. In the second case, the probability is computed based on the difference in fitness values between the new and current solution. Therefore, this acceptance probability $P(t)$is defined by:

$$P(t) = e^{-\frac{\Delta f_t}{T}} \tag{8}$$

Where T is the current temperature (scaling parameter), Δf_t is the difference in the values of the result between the current and the candidate solutions at step. The temperature has an initial value T_0 and it is reduced progressively according to a predefined cooling schedule. The temperature at iteration on t is calculated as follows;

$$T(t+1) = \alpha \cdot T(t); 0 \geq \alpha \leq 1 \tag{9}$$

Where α is called the cooling rate. Note that the purpose of the acceptance probability is prevents the algorithm from getting trapped in a local optimum by allowing non-improving moves. Thus, by allowing the current solution to worsen temporarily with a given probability, simulated annealing is opens up the probability to find the global optimum.

3. RADIUS PARTICLE SWARM OPTIMZATION FUSE WITH SIMULATED ANNEALING

Motivation

The idea of RPSO-SA involves a fusion state and final state as shown in Fig.1. In the fusion state, RPSO-SA is implemented using the temperatures. Thus, the initial temperature is obtained by dividing the difference between the maximum and minimum fitness of the initial particle in the swarm by the acceptance probability. The initial temperature is adaptive in each KP dataset. During fusion state, the RPSO-SA finds the best solution and the proposed algorithm skips local optima by allowing the exploration of the problem space in the direction that leads to a local increase in the next solution. In the final state, after the system is cool, each particle in the swarm updates position using the best position from the fusion state and then opens up to find the global optimum.

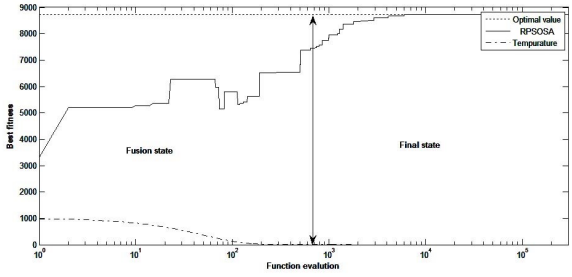

Fig.1. Convergence of RPSO-SA algorithm for MKP

The framework of RPSO-SA uses the strong global search ability of RPSO and the strong local search ability of SA as shown in Algorithm 1. In the RPSOSA, the global best position is selected from agent particles in the search space. Additionally, there are two ways of accepting a new solution. Hence, the RPSO-SA allows the fitness of some particles may be accepting a solution with a worse fitness within a certain probability by the metropolis process of SA.

The algorithm has the advantages of both RPSO and SA algorithms. It solves the KPs with different dimensions and then we will compare the results with both individual RPSO and SA algorithms regarding solution quality and computational efficiency.

Solution representation

To solve the KPs, a candidate solution for KPs represent as the dimension is the number of items *n,* as shown in Fig.2. For the example, we have seven items and after the position of a particle is updated, the position representation is: 0.20, 0.90, 0.10, 0.70, 0.05, 0.65 and 0.80.

j_0	j_1	j_2	j_3	j_4	j_5	j_6
0.20	0.90	0.10	0.70	0.05	0.65	0.80

Fig.2. Representation of the position in each dimension for KPs.

Transfer method

The basic idea of the position update equation for RPSO-SA in the binary domain is taken from position update equation of binary PSO. If a random number value is more than the sigmoid value of the velocity then the position of particle takes the value 0 or 1, as shown in Fig.3. For example, after the position of a particle is updated with the sigmoid function and then compared to the random number, the position representation is: 0, 1, 0, 1, 1, 1 and 1. We then select item number 2, 4, 5, and 6.

$v_{i,j}(t+1)$	0.20	0.90	0.10	0.70	0.05	0.65
$S\left(v_{i,j}(t+1)\right)$	0.55	0.71	0.52	0.67	0.51	0.66
Random number	0.63	0.55	0.97	0.09	0.47	0.26
$x_{i,j}(t+1)$	0	1	0	1	1	1

Fig.3. Representation of transfer of positions for KPs.

4. EXPERIMENTS AND RESULTS

Knapsack test problems

The benchmarks are selected from MP-Test-data SAC-94 suite. A set of 0–1 knapsack instances are taken from [15]. The proposed method is tested using 24 different numbers of items n and knapsack constraints for 0–1KP and nine different problem classes, each with different item/knapsack constraint combinations for MKP. Therefore, the comparison is made on the basis of optimum rate, best profit and average profit.

Parameter setting

The size of swarm is 60, iteration number is 4000 or 240000 function evaluations, x_{max} and v_{max} are equal and within the range of [-4, 4], w is 0.98, the cooling rate α is 0.95 and frozen ε is 0.001.

0–1 KP result

Total of 24 0–1KP test benchmark functions are used to compare the performance of RPSO-SA. Table 1 presents the numerical result of 0–1KP produced by the three algorithms: RPSO-SA, RPSO and SA.

It is obvious that RPSO-SA presents higher optimum rates for all instances. It can also be seen that RPSO-SA is better than RPSO and SA from the point of view of best profit and average profit. In conclusion, RPSO-FA is relatively better than RPSO and SA.

MKP result

For the MKP, the convergence results of RPSO, SA and RPSO-SA are demonstrated in Fig. 4 to 6. It is clearly shown that the RPSO-SA has achieved the optimum solution faster than the other two methods.

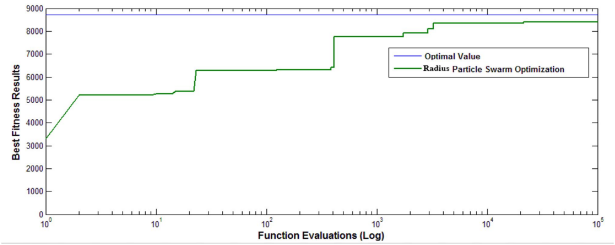

Fig. 4. Convergence of RPSO algorithm for MKP (Sent2.dat)

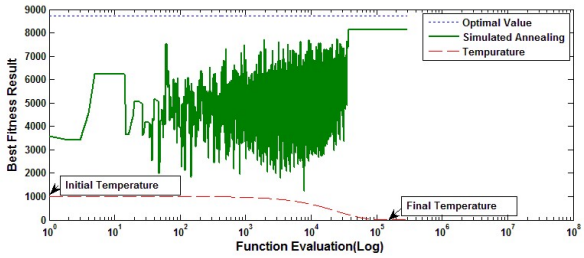

Fig. 5. Convergence of SA Algorithm for MKP (Sent2.dat)

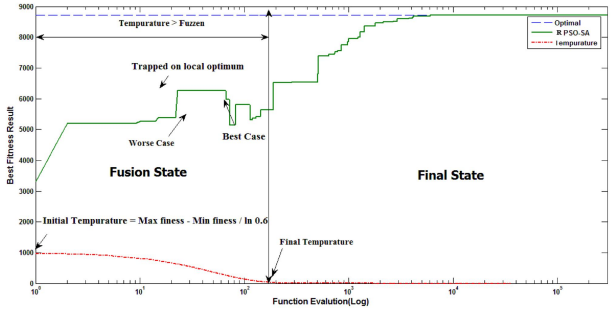

Fig. 6. Convergence of RPSO-SA algorithm for MKP (Sent2.dat)

The MKP numerical results of the instance datasets are illustrated in Table 2. The number of item instances range between 20 and 70. Table 2 reports the detailed results produced by the three algorithms: RPSO-SA, RPSO and SA. It is clear that RPSO-SA is more reliable than RPSO and SA in terms of optimum rate, best and average profit.

5. CONCLUSION

The local optimum is frequently found in the knapsack problems. Author proposed a novel approach to solve the knapsack problems by combining the radius particle swarm optimization (RPSO) and simulated annealing (SA) algorithm. The structure of the method enables the advantages of RPSO which has strong global search ability and SA which has strong local search ability to obtain optimum solution rapidly. The basic idea of the algorithm consists of fusion state and final state. In the fusion state, RPSO-SA is implemented using the temperatures. The initial temperature is the difference between the maximum fitness and minimum fitness of the initial particle in swarm. During fusion state, the RPSO find the best solution and the SA skip local optimum by allowing the exploration of the problem space in directions that lead to a local increase in the next solution. The final state, after the system is cool, each particle in swarm update position by the best position from the fusion state, then opens up to find the global optimum. The RPSO-SA is applied to solve the well-known optimization KPs and MKPs problems. It employs five datasets of the 0-1 knapsack and seven datasets of the multi-dimension knapsack

problem. The experiments have shown that the performance of RPSO-SA outperform the RPSO and SA.

6. REFERENCES

[1]. S. Goyal and A. Parashar, "A Proposed Solution to Knapsack Problem Using Branch & Bound Technique", **Int. J. for Innovative Research in Multidisciplinary Field**, V.2(7), 2016, pp. 240-246.

[2]. A. Rong, and J. R. Figueira, "Computational performance of basic state reduction based dynamic programming algorithms for bi-objective 0–1 knapsack problem", **Computers and Mathematics with Application**s, vol. 63, 2012, pp.1462–1480.

[3]. Z. Ren, Z. Feng , and A. Zhang, "Fusing ant colony optimization with Lagrangian relaxation for the multiple-choice multidimensional knapsack problem", **Information Sciences**, vol. 182, 2012, pp.15–29.

[4]. Mana Anantathanvit and Mudarmeen Munlin, "Using K-means Radius Particle Swarm Optimization for the Travelling Salesman Problem*",* **IETE Technical Review**, vol. 33, 2016, pp. 172-180.

[5]. S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimzation by simulated anneling", **Science**, Vol. 220(4598),1983, pp. 671-680.

[6]. Li Y., He Y., Li H., Guo X., Li Z., "A Binary Particle Swarm Optimization for Solving the Bounded Knapsack Problem", **Computational Intelligence and Intelligent Systems**, 2019, pp. 50-60.

[7]. K. Iwama and G. Zhang, "Online knapsack with resource augmentation", **Information Processing Letters**, Vol. 110(22), 2010, pp. 1016-1020.

[8]. E. Bas, "Surrogate relaxation of a fuzzy multidimensional 0–1 knapsack model by surrogate constraint normalization rules and a methodology for multi-attribute project portfolio selection", **Engineering Application of Artificial Intellience**, Vol. 25(5), 2012, pp. 958-970.

[9]. M.M. Baldi, G. Perboli and R. Tadei, "The three-dimensional knapsack problem with balancing constraints"*,* **Applied Mathematics and Computation***,* V.218, 2012, pp. 9802–9818.

[10]. A. Pham, "The improvent knapsack chiper", **Computer Comunications**, Vol. 34(3), 2011, pp. 342-343.

[11]. B.Thiongane, A. Nagih, and G. Plateau, "Lagrangean heuristics combined with reoptimization for the 0–1 bidimensional knapsack problem", **Discrete Applied Mathematics**, Vol. 154(15), 2006, pp. 2200–2211.

[12]. J. C. Bansal and K. Deep, "A Modified Binary Particle Swarm Optimization for Knapsack Problems", **Applied Mathematics and Computation**, Vol. 218(22), 2012, pp.11042–11061.

[13]. X. Zhang, S. Huang, Y. Hu, Y. Zhang, S. Mahadevan, and Y. Deng, "Solving 0-1 knapsack problems based on amoeboid organism algorithm", **Applied Mathematics and Computation**, Vol. 219(19), 2013, pp. 9959–9970.

[14]. L.Wang, S.Wang, and Y. Xu, "An effective hybrid EDA-based algorithm for solving multidimensional knapsack problem", **Expert Systems with Applications**, Vol. 39(5), 2012, pp. 5593–5599.

[15]. MP-Test-Data SAC-94 [Online] Avaliable: http://elib.zib.de/pub/mp-testdata/ip/sac94-suite/index.html

Algorithm 1 The framework of RPSO-SA for KPs.

1. ***//Step (1) Initialization***
2. **Set** the parameters including swarm size, dimension size, t, t_{max}, v_{min}, v_{max}, x_{min}, x_{max}, w, μ, c_1, c_2 temperatures (T), cooling rate (α) and frozen (ε)
3. **Initialise** each dimension of gbest position with x_{min}
4. **Calculate** the gbest value of gbest position with the fitness function
5. **Initialise** each particle with random position and velocity
6. **for** each particle in the swarm **do**
7. **Calculate** the fitness value of particle with the fitness function
8. **Update** its pbest value with its fitness value
9. **Assign** its position to its pbest position
10. **if** its pbest value **better than** the gbest value **then**
11. **Update** the gbest value with its fitness value
12. **Assign** its position to the gbest position
13. **end if**
14. **end for**
15. ***//Step (2) Reproduction and updating loop***
16. ***t = 0***
17. temperatures $(T) = \frac{maximum\ fitness\ value\ of\ particle - minimum\ \ fitness\ value\ of\ particle}{\ln 0.6}$
18. **for** each iteration t to t_{max} **do**
19. **for** each particle in the swarm **do**
20. **for** each dimension in the particle **do**
21. **Calculate** the velocity by equation(3.4)
22. **if** its velocity value more than v_{max}

Update its velocity value with v_{max}
else if its velocity value less than v_{min}
Update its velocity value with v_{min}
end if
Update the position by equation (3.5)
if its position value more than x_{max}
Update its position value with x_{max}
else if its position value less than x_{min}
Update its position value with x_{min}
end if
end for
Calculate the fitness value of particle with the fitness function
if its fitness value **better than** its pbest value **then**
Update its pbest value with its fitness value
Assign its position to its pbest position
end if
//***Step (2.1) Find the agent particle on the radius-neighbourhood***
Define the radius-neighbourhood by equation 3.1 and equation) 3.2(
Find the abest particle in the radius swarm group by equation)3.3(
Step (2.2) Find the gbest on the global neighbourhood
for each abest particle **do**
if fitness value of abest particle **better than** the gbest value **then**
Update the gbest value with the fitness value of abest particle
Assign abest position to the gbest position
else if (temperatures (T) is more than frozen (ε))
//Step (2.3) SA algorithm
Assign the gbest position to current solution
Generate the candidate solution on the current solution
$\Delta f = \text{the finess value of candidate solution} - \text{the finess value of current solution}$
if the fitness value of candidate solution **better than** the fitness value of current
Update the gbest value with the fitness value of candidate solution
else if $\text{Random}[0.1] > e^{-\frac{\Delta f}{T}}$ **then**
Update the gbest value with the fitness value of candidate solution
Assign the candidate solution to the gbest position
end if
end if
end for
end for
$T = \alpha . T$
end for

Table 1. The 0–1KP results.

Dataset	Items	OPT.	Algorithm	Optimum rate (%)	Profit	
					Best	Average
ks 8b	8	3813669	**RPSO-SA**	**100**	**3813669**	**3813669**
			RPSO	100	3813669	3813669
			SA	100	3813669	3813669
ks 12b	12	649859	**RPSO-SA**	**100**	**649859**	**649859**
			RPSO	85	649859	647645
			SA	66	649859	627645
ks 16b	16	9352998	**RPSO-SA**	**96**	**9352998**	**9352818**
			RPSO	92	9352998	9352601
			SA	74	9352998	9341252
ks 20b	20	9818261	**RPSO-SA**	**88**	**9818261**	**9792115**
			RPSO	86	9818261	9699235
			SA	72	9818261	9655125
ks 24b	24	12233713	**RPSO-SA**	**70**	**12233713**	**12216251**
			RPSO	65	12233713	12125120
			SA	30	12233713	11245230

Table 2. The MKP results.

Dataset	m	n	Opt.	Algorithm	Optimum rate (%)	Profit	
						Best	Average
Pet2	10	10	87061	**RPSO-SA**	**100**	**87061**	**87061**
				RPSO	58	87061	83369
				SA	0	83369	80312
Pb4	2	29	95168	**RPSO-SA**	**87**	**95168**	**94002**
				RPSO	40	95168	90900
				SA	0	94801	90809
Pb5	10	20	2139	**RPSO-SA**	**81**	**2139**	**2034**
				RPSO	70	2139	2002
				SA	0	2088	1986
Pb6	30	40	776	**RPSO-SA**	**68**	**776**	**680**
				RPSO	15	776	652
				SA	0	686	642
Sent1	30	60	7772	**RPSO-SA**	**65**	**7772**	**7723**
				RPSO	34	7772	7758
				SA	0	6939	6272
Sent2	30	60	8722	**RPSO-SA**	**42**	**8722**	**8512**
				RPSO	5	8722	8112
				SA	0	8311	7995
Weish20	5	70	9450	**RPSO-SA**	**89**	**9450**	**9206**
				RPSO	60	9450	9014
				SA	0	7787	7078