A Branch and Bound-PSO Hybrid Algorithm for Solving Integer Separable Concave Programming Problems

Yuelin Gao\textsuperscript{2,3}, Zihui Ren\textsuperscript{4} and Chengxian Xu\textsuperscript{2}

\textsuperscript{2} School of Finance and Economics, Xi’an Jiaotong University
Xi’an, 710049, China
mxxu@mail.xjtu.edu.cn

\textsuperscript{3} Department of Information and Computation Science
Northwest Second National College
Yin Chuan 750021, China
gaoyuelin@263.net

\textsuperscript{4} School of Mathematics and Computer
Ningxia University, Yin Chuan 750021, China
renzihui@etang.com

Abstract

A branch and bound-PSO hybrid algorithm for solving integer separable concave programming problems is proposed, in which the lower bound of the optimal value was determined by solving linear programming relax and the upper bound of the optimal value and the best feasible solution at present were found and renewed with particle swarm optimization (PSO). It is shown by the numerical results that the branch and bound-PSO hybrid algorithm is better than the branch and bound algorithm in the computational scale and the computational time and the computational precision and overcome the convergent difficulty of particle swarm optimization (PSO).

Keywords: Integer programming; separable concave programming; branch and bound method; particle swarm optimization; linear programming relax

\textsuperscript{1}The work is supported by the Foundations of National Natural Science in Ningxia (grants NZ0676) and Post Doctor of China(20060401001), and by the Science Research Projects of Ministry of Education of China(06JA630056), and Ningxia’s Colleges and Universities in 2005.
1 Introduction

Consider the integer separable concave programming problem below:

\[
\begin{align*}
\text{(ISCCP)} & \quad \min \phi(x) = \sum_{j=1}^{n} f_j(x_j) \\
\text{s.t.} & \quad x \in D = \{x \in \mathbb{R}^n | Ax \leq b\}, \\
& \quad x \in \mathbb{Z}^n.
\end{align*}
\]

where each \( f_j(x_j) \) is a concave function over \( \mathbb{R} \), \( \mathbb{Z}^n \) is a set which consists of all \( n \)-dimension integer points; \( A = (a_{ij})_{n \times m} \in \mathbb{R}^{m \times n}, b = (b_1, b_2, \ldots, b_m)^T \in \mathbb{R}^m \), the feasible field \( D \) is a bounded convex polyhedron.

Problem (ISCCP) are encountered in a variety of areas, such as capital budgeting [2], computer-aided layout design [5], portfolio selection [3], site selection for electric message systems [6] and shared fixed costs [7], etc. The methods for solving the problem (ISCCP) has mainly method of dynamic programming, branch and bound method, the method of computational intelligence [1,4,8-10].

In the paper, we present a branch and bound-PSO hybrid algorithm for solving the problem (ISCCP), in which the lower bound of the optimal value was determined by solving linear programming relax and the upper bound of the optimal value and the best feasible solution at present were found and renewed with particle swarm optimization (PSO). It is shown by the numerical results that the branch and bound-PSO hybrid algorithm is better than the branch and bound algorithm (BBA) and overcome the convergent difficulty of particle swarm optimization (PSO).

Section 2 gives a good linear programming relax of the problem (ISCCP). Section 3 gives a particle swarm optimization algorithm based on the penalty function for solving the problem (ISCCP) to find and renew the best feasible solution of the problem (ISCCP) and the upper bound of the optimal value of it at present. Section 4 describes a branch and bound-PSO hybrid algorithm. Section 5 gives several numerical examples to show that the proposed algorithm is effective.

2 A linear programming relaxed approximation

Firstly, we give the continuous relaxed programming of the problem (ISCCP)

\[
\begin{align*}
\min \phi(x) &= \sum_{j=1}^{n} f_j(x_j) \\
\text{s.t.} & \quad x \in D = \{x \in \mathbb{R}^n | Ax \leq b\}, \\
& \quad x \in [l, u].
\end{align*}
\]
where \([l, u]\) is a rectangle over \(\mathbb{R}^n\) which contains \(D\). Because \(D\) is a bounded convex polyhedron, so we can find a rectangle \([l, u] \subseteq \mathbb{R}^n\) such that \(D \subseteq [l, u]\) by solving linear programming.

Because each \(f_j(x_j)\) is a concave function over \(\mathbb{R}\), so its convex envelope over \([l_j, u_j]\) is a line through two points \((l_j, f_j(l_j))\) and \((u_j, f_j(u_j))\), i.e.

\[
\text{lb}_j(x_j) = \frac{(f_j(u_j) - f_j(l_j))}{u_j - l_j}(x - l_j) + f_j(l_j),
\]

thus we can obtain the best linear programming relaxed approximation of the problem (ISCCP) below:

\[
\begin{align*}
\min & \quad \phi(x) = \sum_{j=1}^{n} \text{lb}_j(x_j) \\
\text{subject to} & \quad x \in D = \{x \in \mathbb{R}^n | Ax \leq b\}, \\
& \quad x \in [l, u].
\end{align*}
\]

The optimal value of the problem (3) is a lower bound of the optimal value of the problem (ISCCP).

3 The particle swarm optimization algorithm of integer programming

The particle swarm optimization algorithm (PSO)[8, 9, 10] is a kind of computational intelligent technology which is put forward by Kennedy and Eberhart in 1995 and has global optimal property, but PSO’s convergence is not proved.

Below is a penalty function of the problem (ISCCP):

\[
p(x) = \sum_{j=1}^{n} f_j(x_j) + M \left\{ \sum_{i=1}^{m} \min \{0, b_k - \sum_{j=1}^{n} a_{ij}x_j\} \right\}
\]

where \(M > 0\) is a penalty parameter.

We now give a PSO algorithm based on the penalty function (4) for solving the problem (ISCCP). \(N_c\) is noted as the iteration times when the PSO algorithm stops, \(M_c\) is noted as the number of the particles in a particle swarm, \(p_{sb}\) is noted as the best position by which a particle has gone so far and \(p_{gb}\) is noted as the best position by which all the particles in the particle swarm has gone so far as well as \(x_{gb}\) is noted as the best feasible position by which all the particles in the particle swarm has gone so far. \(V_{max}^{i}\) is noted as the biggest velocity of the particle \(x_i (i = 1, 2, \cdots, M_c)\).

PSO algorithm based on the penalty function
Step 1. Set $t = 1, M = 1000, Nc = 100, Mc = 60$. Produce randomly a particle swarm, the initial position of each particle $x_i (i = 1, 2, \cdots, Mc)$ in which is $x_{ij}(0), j = 1, 2, \cdots, n$ and the initial velocities $v_{ij}(0), j = 1, 2, \cdots, n$. Compute the fitness of each particle. Determine $p_{sb}$ of each particle, and $p_{gb}$ and $x_{gb}$ of all the particles in the particle swarm.

Step 2. Set $t = t + 1, M = M \times t$. For each particle $x_i (i = 1, 2, \cdots, Mc)$, from the iteration formula below:

$$
\begin{align*}
  v_{ij} &= \omega v_{ij} + c_1 rand_1 (p_{ij} - x_{ij}) + c_2 rand_2 (p_{gb} - x_{ij}) \\
  x_{ij} &= x_{ij} + v_{ij}, j = 1, 2, \cdots, n.
\end{align*}
$$

(5)

where $w \in [0, 1.2]$ is inertia weight, $c_1 = 2$ and $c_2 = 1.7$ are acceleration constants, $rand_1$ and $rand_2$ are two random functions over $[0, 1]$. If $v_{ij} > V_{max}^i$ in (5), then $v_{ij} = V_{max}^i (j = 1, 2, \cdots, n)$.

Step 3. For each particle, compute $p_{sb}$. For the particle swarm, compute $p_{gb}$ and $x_{gb}$.

Step 4. If $t = Nc$, then outcome $x_{opt} = x_{gb}$, stop; else go to step 2.

All the coefficients in the PSO algorithm are determined through the numerical test in Section 5 and the PSO algorithm can find better feasible solution and better upper bound.

4 Description of a branch and bound-PSO hybrid algorithm

In this section, we describe a branch and bound-PSO hybrid algorithm for solving the problem (ISCCP), in which the branching procedure is usual integer rectangle two-partitioning one and the bounding lower procedure is by solving the problem (3) as well as in the bounding upper procedure the PSO algorithm in Section 3 is used. Denote that $R = [l, u]$.

We now describe the branch and bound-PSO hybrid algorithm (BB-PSO-HA).

Step 0. (Initialization) Set $k = 0, \Omega = \{R\}$. Determine the best lower LB at present by solving the problem (3) and the best upper bound UB and the best feasible solution $xx_{best}$ of the problem (ISCCP) at present with PSO algorithm.

Step k ($k = 1, 2, \cdots$)

k1. (Termination) If $\Omega = \Phi$ or $\frac{UB - LB}{UB} < eps$, then outcome $xx_{best}$ and $OPT = UB$.

k2. (Selection Rule) Find a subrectangle $R_k$ in $\Omega$ such that $LB(R_k) = LB$.

k3. (Branching) Partition $R_k$ into two subrectangles $R_{k+1,1}$, $R_{k+1,2}$, and each subrectangle is reduced into integer rectangle each vertex point of which is integer point. The obtained two integer subrectangles are noted as $R_{k+1,1}$ and $R_{k+1,2}$ too. Set $\Omega = (\Omega - R_k) \cup \{R_{k+1,1}, R_{k+1,2}\}$.
A branch and bound-PSO hybrid algorithm

k4. (Lower Bounding) Solve the problem (3) over \( R_{k+1,1} \) and over \( R_{k+1,2} \) respectively to obtain new lower \( LB \).

k5. (Upper Bounding) Renew \( UB \) and \( xx_{best} \) with the PSO algorithm in Section 3.

k6. (Deleting Rule) Set \( \Omega = \Omega - \{ R \in \Omega : LB(R) \geq UB \} \) and \( k = k + 1 \). Go to k1.

5 Numerical Computation

We solve the three problems (6)-(8) below with BBA and BB-PSO-HA:

\[
\begin{align*}
\text{min} & \quad \sum_{i=1}^{n} (c_i x_i - d_i x_i^2) \\
\text{subject to} & \quad \sum_{i=1}^{n} a_i x_i \leq b, \\
& \quad x \in [-2, 4], x \in Z, \\
& \quad i = 1, 2, \ldots, n.
\end{align*}
\]

(6)

where \( c_i \in [10, 20], d_i \in [10, 20], a_i \in [0, 50], b = 3.8 \sum_{i=1}^{n} a_i \).

\[
\begin{align*}
\text{min} & \quad \sum_{i=1}^{n} \log(c_i x_i + d_i) \\
\text{subject to} & \quad \sum_{i=1}^{n} a_i x_i \leq b, \\
& \quad x \in [1, 20], x \in Z, \\
& \quad i = 1, 2, \ldots, n.
\end{align*}
\]

(7)

where \( c_i \in [10, 20], d_i \in [10, 20], a_i \in [0, 50], b = 1.2 \sum_{i=1}^{n} a_i \).

\[
\begin{align*}
\text{min} & \quad \sum_{i=1}^{n} (c_i x_i + \frac{1}{x_i^2}) \\
\text{subject to} & \quad \sum_{i=1}^{n} a_i x_i \leq b, \\
& \quad x \in [1, 6], x \in Z, \\
& \quad i = 1, 2, \ldots, n.
\end{align*}
\]

(8)

where \( c_i \in [-9, 9], d_i \in [1, 7], a_i \in [0, 50], b = 3.8 \sum_{i=1}^{n} a_i \).

The procedure of BBA and BB-PSO-HA are compiled with Matlab7.0.1 in person computer on DELL-P4-Intel865-512MB. We produce randomly twenty examples for the problems (6)-(8) in \( n = 60, 100, 150, 200, 300, 500, 800, 1000, 1500, 2000, 3000, 4000 \) respectively and solve these examples with BBA and BB-PSO-HA respectively. The results of the numerical computation can be seen at Table1-Table6. "Iteration" and "Cputime" are noted as the iteration.
times and computational time respectively. "Avg, Max, Min" are noted as the iteration times and computational time of "average, maximum, minimum" respectively. We can say that the computation fails if BB-PSO-HA or BBA does not stop when iteration=10000.

It is shown by the numerical results from Table 1-Table 6 that BB-PSO-HA is better than BBA in the computational scale and the computational time and the computational precision.

<table>
<thead>
<tr>
<th>n</th>
<th>Avg</th>
<th>Max</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>MIN</th>
<th>failure rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>7000</td>
<td>10000</td>
<td>1</td>
<td>472.2258</td>
<td>1035.8</td>
<td>0.0940</td>
<td>0.05</td>
</tr>
<tr>
<td>100</td>
<td>7580</td>
<td>10000</td>
<td>1</td>
<td>674.9794</td>
<td>1331.5</td>
<td>0.0780</td>
<td>0.05</td>
</tr>
<tr>
<td>150</td>
<td>7211</td>
<td>10000</td>
<td>1</td>
<td>844.7176</td>
<td>2574.9</td>
<td>0.1560</td>
<td>0.05</td>
</tr>
<tr>
<td>200</td>
<td>6776</td>
<td>10000</td>
<td>1</td>
<td>840.5947</td>
<td>3206.8</td>
<td>0.2970</td>
<td>0.05</td>
</tr>
<tr>
<td>300</td>
<td>8366</td>
<td>10000</td>
<td>1</td>
<td>1793.9932</td>
<td>5450.0</td>
<td>0.2030</td>
<td>0.10</td>
</tr>
<tr>
<td>500</td>
<td>6298</td>
<td>10000</td>
<td>3</td>
<td>2405.8687</td>
<td>8278.6</td>
<td>0.6400</td>
<td>0.10</td>
</tr>
<tr>
<td>800</td>
<td>5288</td>
<td>10000</td>
<td>2</td>
<td>4491.6621</td>
<td>8611.0</td>
<td>0.9850</td>
<td>0.083</td>
</tr>
<tr>
<td>1000</td>
<td>4357</td>
<td>10000</td>
<td>432</td>
<td>4143.4605</td>
<td>22135.0</td>
<td>181.0780</td>
<td>0.083</td>
</tr>
</tbody>
</table>

Table 1 Numberical results for the problem 1

<table>
<thead>
<tr>
<th>n</th>
<th>Avg</th>
<th>Max</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>MIN</th>
<th>failure rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>81</td>
<td>10000</td>
<td>1</td>
<td>584.3867</td>
<td>87775</td>
<td>9.090</td>
<td>0.05</td>
</tr>
<tr>
<td>100</td>
<td>95</td>
<td>648</td>
<td>1</td>
<td>795.3813</td>
<td>6032.4</td>
<td>12.1100</td>
<td>0</td>
</tr>
<tr>
<td>150</td>
<td>47</td>
<td>10000</td>
<td>1</td>
<td>1089.8666</td>
<td>9786.0</td>
<td>21.8590</td>
<td>0.2</td>
</tr>
<tr>
<td>180</td>
<td>122</td>
<td>10000</td>
<td>1</td>
<td>1462.1903</td>
<td>12588</td>
<td>21.6560</td>
<td>0.09523</td>
</tr>
<tr>
<td>200</td>
<td>178</td>
<td>10000</td>
<td>1</td>
<td>1510.5300</td>
<td>22573</td>
<td>29.0150</td>
<td>0.05</td>
</tr>
<tr>
<td>300</td>
<td>60</td>
<td>197</td>
<td>1</td>
<td>2100.8955</td>
<td>34889</td>
<td>36.0150</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>28</td>
<td>124</td>
<td>1</td>
<td>6908.9045</td>
<td>90178</td>
<td>72.7340</td>
<td>0</td>
</tr>
<tr>
<td>800</td>
<td>80</td>
<td>10000</td>
<td>1</td>
<td>959.0601</td>
<td>10874</td>
<td>116.3750</td>
<td>0.15</td>
</tr>
<tr>
<td>1000</td>
<td>20</td>
<td>10000</td>
<td>1</td>
<td>1991.250</td>
<td>18286</td>
<td>147.516</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Table 3: Numerical results for the problem 2

<table>
<thead>
<tr>
<th>Ex2</th>
<th>$Eps = 10^{-5}$</th>
<th>BBA</th>
<th>Iteration</th>
<th>Cputime(Seconds)</th>
<th>failure rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>60</td>
<td>1076</td>
<td>5005</td>
<td>1</td>
<td>49.9905</td>
<td>325.8750</td>
</tr>
<tr>
<td>100</td>
<td>1579</td>
<td>10000</td>
<td>1</td>
<td>133.3667</td>
<td>874.7650</td>
</tr>
<tr>
<td>150</td>
<td>1655</td>
<td>9988</td>
<td>1</td>
<td>270.3304</td>
<td>2408.5</td>
</tr>
<tr>
<td>200</td>
<td>1875</td>
<td>10000</td>
<td>1</td>
<td>520.5757</td>
<td>4111.1</td>
</tr>
<tr>
<td>300</td>
<td>1081</td>
<td>8876</td>
<td>1</td>
<td>296.6009</td>
<td>2549.9</td>
</tr>
<tr>
<td>500</td>
<td>2633</td>
<td>10000</td>
<td>1</td>
<td>1054.8215</td>
<td>4822.2</td>
</tr>
<tr>
<td>800</td>
<td>558</td>
<td>10000</td>
<td>2</td>
<td>484.9714</td>
<td>8855.0</td>
</tr>
<tr>
<td>1000</td>
<td>780</td>
<td>9888</td>
<td>432</td>
<td>490.8955</td>
<td>8402.8</td>
</tr>
</tbody>
</table>

Table 4: Numerical results for the problem 2

<table>
<thead>
<tr>
<th>Ex2</th>
<th>$Eps = 10^{-5}$</th>
<th>BBA-PSO</th>
<th>Iteration</th>
<th>Cputime(Seconds)</th>
<th>failure rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>60</td>
<td>25</td>
<td>166</td>
<td>1</td>
<td>274.9855</td>
<td>1814.8</td>
</tr>
<tr>
<td>100</td>
<td>8</td>
<td>75</td>
<td>1</td>
<td>142.8922</td>
<td>1488.4</td>
</tr>
<tr>
<td>150</td>
<td>16</td>
<td>164</td>
<td>1</td>
<td>449.3077</td>
<td>4546.0</td>
</tr>
<tr>
<td>180</td>
<td>11</td>
<td>175</td>
<td>1</td>
<td>171.9088</td>
<td>2379.8</td>
</tr>
<tr>
<td>200</td>
<td>18</td>
<td>160</td>
<td>1</td>
<td>635.5753</td>
<td>5797.7</td>
</tr>
<tr>
<td>300</td>
<td>14</td>
<td>178</td>
<td>1</td>
<td>451.2245</td>
<td>3947.8</td>
</tr>
<tr>
<td>500</td>
<td>15</td>
<td>144</td>
<td>1</td>
<td>1017.3059</td>
<td>9394.1</td>
</tr>
<tr>
<td>800</td>
<td>4</td>
<td>10000</td>
<td>1</td>
<td>594.2683</td>
<td>6732.6</td>
</tr>
<tr>
<td>1000</td>
<td>18</td>
<td>10000</td>
<td>1</td>
<td>3493.1909</td>
<td>50020</td>
</tr>
<tr>
<td>1500</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>297.8065</td>
<td>1003.2</td>
</tr>
<tr>
<td>2000</td>
<td>5</td>
<td>50</td>
<td>1</td>
<td>3057.3791</td>
<td>41250</td>
</tr>
</tbody>
</table>

Table 5: Numerical results for the problem 3

<table>
<thead>
<tr>
<th>Ex3</th>
<th>$Eps = 10^{-5}$</th>
<th>BBA</th>
<th>Iteration</th>
<th>Cputime(Seconds)</th>
<th>failure rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>60</td>
<td>2543</td>
<td>10000</td>
<td>1</td>
<td>291.1397</td>
<td>680.0630</td>
</tr>
<tr>
<td>100</td>
<td>2942</td>
<td>10000</td>
<td>1</td>
<td>248.8008</td>
<td>1575.8</td>
</tr>
<tr>
<td>150</td>
<td>3765</td>
<td>10000</td>
<td>1</td>
<td>348.9752</td>
<td>2768.2</td>
</tr>
<tr>
<td>200</td>
<td>4491</td>
<td>10000</td>
<td>1</td>
<td>615.3112</td>
<td>2922.6</td>
</tr>
<tr>
<td>300</td>
<td>2588</td>
<td>10000</td>
<td>1</td>
<td>704.9035</td>
<td>4100.5</td>
</tr>
<tr>
<td>500</td>
<td>919</td>
<td>10000</td>
<td>1</td>
<td>161.2608</td>
<td>1721.5</td>
</tr>
<tr>
<td>800</td>
<td>2303</td>
<td>10000</td>
<td>2</td>
<td>664.1810</td>
<td>2634.0</td>
</tr>
<tr>
<td>1000</td>
<td>8563</td>
<td>10000</td>
<td>1</td>
<td>1171.1894</td>
<td>4193.9</td>
</tr>
</tbody>
</table>
Table 6 Numerical results for example 3

<table>
<thead>
<tr>
<th>Ex3</th>
<th>BBA-PSO</th>
<th>Iteration</th>
<th>Cputime(Seconds)</th>
<th>failure rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eps</td>
<td></td>
<td>n</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>32</td>
<td>526</td>
<td>1</td>
</tr>
<tr>
<td>150</td>
<td></td>
<td>2</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>180</td>
<td></td>
<td>28</td>
<td>463</td>
<td>1</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td>22</td>
<td>374</td>
<td>1</td>
</tr>
<tr>
<td>300</td>
<td></td>
<td>12</td>
<td>197</td>
<td>1</td>
</tr>
<tr>
<td>500</td>
<td></td>
<td>10</td>
<td>10000</td>
<td>1</td>
</tr>
<tr>
<td>800</td>
<td></td>
<td>3</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>1500</td>
<td></td>
<td>3</td>
<td>10000</td>
<td>1</td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3000</td>
<td></td>
<td>7</td>
<td>38</td>
<td>1</td>
</tr>
<tr>
<td>4000</td>
<td></td>
<td>24</td>
<td>37</td>
<td>1</td>
</tr>
</tbody>
</table>

References


Received: September 30, 2006
کتاب های به زبان انگلیسی

- Particle Swarm Optimization: Theory, Techniques and Applications
  ترجمه عنوان: پارسی مبتکر
  مولف: Andrea E. Olsson
  سال: 2011
  انتشارات: Nova Science Pub Inc
  لینک دسترسی:

- Evolutionary Optimization Algorithms
  ترجمه عنوان: الگوریتم های تولیدی
  مولف: Dan Simon
  سال: 2008
  انتشارات: Wiley
  لینک دسترسی:

- Computational Intelligence: An Introduction
  ترجمه عنوان: هوش محاسباتی
  مولف: Andries P. Engelbrecht
  سال: 2007
  انتشارات: Wiley
  لینک دسترسی:

- Nature-Inspired Optimization Algorithms
  ترجمه عنوان: الگوریتم های الهام گرفته از طبیعت
  مولف: Xin-She Yang
  سال: 2014
  انتشارات: Elsevier
  لینک دسترسی:

- Introduction to Mathematical Optimization: From Linear Programming to Metaheuristic
  ترجمه عنوان: بهینه سازی خطی تا الگوریتم فرا
  مولف: Xin-She Yang
  سال: 2008
  انتشارات: Cambridge International Science Publish
  لینک دسترسی:

- Metaheuristic Applications
  ترجمه عنوان: الگوریتم های فرا کاربردی
  مولف: Xin-She Yang
  سال: 2010
  انتشارات: Wiley
  لینک دسترسی:

- Particle Swarm Optimization
  ترجمه عنوان: فراکریز
  مولف: Maurice Clerc
  سال: 2006
  انتشارات: Wiley-ISTE
  لینک دسترسی:

- Metaheuristics: From Design to Implementation
  ترجمه عنوان: الگوریتم های فرا کاربردی
  مولف: El-Ghazali Talbi
  سال: 2009
  انتشارات: Wiley
  لینک دسترسی:

- Introduction to Mathematical Optimization: From Linear Programming to Metaheuristic
  ترجمه عنوان: بهینه سازی خطی تا الگوریتم فرا
  مولف: Xin-She Yang
  سال: 2008
  انتشارات: Cambridge International Science Publish
  لینک دسترسی:

- Metaheuristic Applications
  ترجمه عنوان: الگوریتم های فرا کاربردی
  مولف: Xin-She Yang
  سال: 2010
  انتشارات: Wiley
  لینک دسترسی:

- Evolutionary Optimization Algorithms
  ترجمه عنوان: الگوریتم های تولیدی
  مولف: Dan Simon
  سال: 2008
  انتشارات: Wiley
  لینک دسترسی:

- Computational Intelligence: An Introduction
  ترجمه عنوان: هوش محاسباتی
  مولف: Andries P. Engelbrecht
  سال: 2007
  انتشارات: Wiley
  لینک دسترسی:

- Nature-Inspired Optimization Algorithms
  ترجمه عنوان: الگوریتم های الهام گرفته از طبیعت
  مولف: Xin-She Yang
  سال: 2014
  انتشارات: Elsevier
  لینک دسترسی:

- Introduction to Mathematical Optimization: From Linear Programming to Metaheuristic
  ترجمه عنوان: بهینه سازی خطی تا الگوریتم فرا
  مولف: Xin-She Yang
  سال: 2008
  انتشارات: Cambridge International Science Publish
  لینک دسترسی: