An exact algorithm for the shortest path problem with position-based learning effects ☆

Yamin Wang[a,b], Xiaoping Li[a,b], Rubén Ruiz[c]

[a] *School of Computer Science and Engineering, Southeast University, Nanjing 211189, China*
[b] *Key Laboratory of Computer Network and Information Integration, Ministry of Education, Nanjing, 211189, China*
[c] *Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46021, València, Spain.*

Abstract

The shortest path problems (SPPs) with learning effects (SPLE) are widespread in practical applications and have not been studied yet. In this paper, we show that learning effects make SPLEs completely different from SPPs. An adapted A* (AA*) is proposed for the SPLE problem under study. Though global optimality implies local optimality in SPPs, it is not true in SPLEs. Because all sub-paths of potential shortest solution paths need to be stored during the search process, a search graph is adopted by AA* rather than a search tree used by A*. Admissibility of AA* is proven. Monotonicity and consistency of the heuristic functions of AA* are redefined and the corresponding properties are analyzed. Consistency/monotonicity relationships between the heuristic functions of AA* and those of A* are explored. Their impacts on efficiency of searching procedures are theoretically analyzed and experimentally evaluated.

Keywords: A* search, Learning effect, Shortest path, Admissibility

1. Introduction

Shortest Path Problems (SPP for short) are widespread in practical applications (e.g., logistics, transportation, robot path planning [1] [2], vehicle routing [3]) and no-wait flow shop scheduling [4]. SPP tries to find the shortest path from the source node to the sink node in a graph. Generally, the distance, time or price of traversing of each arc is called cost. There are a large number of paths in the graph. The shortest path is the one has the minimum total cost. Traditionally, costs of all arcs are assumed to be known in advance and the SPP was called SSPP (Static Shortest Path Problem) [3]. However, there are a lot of DSPPs (Dynamic Shortest Path Problem) in practical systems in which the cost of each arc changes with some factors (e.g., traffic status, learning experiences). Though there are a lot of studies on DSPPs (especially in traffic systems), the DSPP problem with learning experiences or effects (SPLE for short) has not been considered yet.

☆This work is supported by the National Natural Science Foundation of China (Grants 61572127, 61272377) and the Specialized Research Fund for the Doctoral Program of Higher Education (20120092110027). Rubén Ruiz is supported by the Spanish Ministry of Economy and Competitiveness, under the project “RESULT - Realistic Extended Scheduling Using Light Techniques ”(No. DPI2012-36243-C02-01) financed with FEDER funds.

In practice, costs of arcs in a path usually change with learning experience or "learning effect"[5]. "Learning effect "was first observed by Wright [6]. Nowadays there are a lot of topics associating with learning effects [7–9]. The shortest path problems in robot soccer matches (robot space exploration, robot rescue in hostile environments, etc.) are typical DSPP problems with learning experiences in which robots obtain learning experiences by interaction with environments using reinforcement learning [10–12]. More experiences imply shorter possible paths robots can find. In logistic systems, there are a lot of items to be sent to different distribution centers. Finding optimal paths for all items is a typical SPP problem. Post-persons become more and more experienced after they do the pick-up and drop-down operations many times [13], which makes the costs (times for transporting items) change with the learning effects. The no wait flowshop scheduling problem (NWFS) is another typical example. The processing time of a job becomes shorter if it is scheduled later in a sequence because the worker is more and more proficient to setup, clean, operate, control, or maintain machines. This problem can be transferred into the traveling salesman problem (TSP) with learning effects [14], a special case of the SPLE problem.

Generally, there are three types of learning effect models: position-based, sum-of-processing-time-based and experience-based [15]. Position-based learning means that learning is affected by the number of arcs being processed or traversed (the position in a sequence). Sum-of-processing-time-based learning takes into account the total time of all traversed arcs while experience-based learning is dependent on the experience of the processor [15]. These models are suitable for different settings. Position-based learning assumes that learning takes place by processing time independent operations like setting up of machines in scheduling problems. Sum-of-processing-time-based learning is the case where running the press itself is a highly complicated and error-prone process which exists in highly customized products, the production of high-end electric tools, maintenance of airplanes, pimping cars [16]. Experience-based learning describes processing times by "S"-shaped functions which includes three phases: the incipient (start-up) phase, the learning phase and the maturity phase [15]. In this paper, the commonly considered position-based learning effect is considered. The cost on each arc in SPLE is regularly changed with its position in the path.

Among existing methods for solving SPPs, A* is one of the most popular algorithms. The A* algorithm was originally presented by Hart et. al. [17], which was extended from the Dijkstra algorithm [18]. Heuristics are key to the time performance of A*. The A* algorithm usually outperforms other traditional exact algorithms for SPPs [19]. Recently, many variants of A* have been presented for SSPPs, such as NAMOA* (New Approach to Multi-Objective) [20], EES (Explicit Estimation Search) [21], and SSiPP (Short-Sighted Probabilistic Planner) [22]. Though there are many A* algorithms for DSPPs, most of them are for irregular ones (i.e., arc costs change stochastically). The one-to-all DSPP (finding the shortest paths between a start node to all the other nodes in a graph) for a given departure time can be transformed into a SSPP [23]. However, the transformation works only if the FIFO (first-in-first-out) property is satisfied [24]. In the literature [19], adaptations of the A* algorithm have been presented for computing the fastest paths in deterministic discrete-time dynamic networks, which also satisfy the FIFO property. By reusing the preceding searching information to find the shortest paths of a series of similar problems, some incremental versions of A* were proposed, such as LPA* (Lifelong Planning A*) [25], GLPA* (Generalized LPA*) [26], FSA* (Fringe-Saving A*) [27], D* (Dynamic A*) [28], D* Lite [29], and Focussed D* [30]. LPA* [25] uses consistent heuristics. GLPA* [26] is a generalized framework from LPA*. When an A* search for the current search problem deviates from the A* search for the immediately preceding search problem, FSA* [27] restores the content of the OPEN list of A* in time at the point. Based on LPA*, D* Lite [29] is developed, which is simple, easy to analyze, and extendible in multiple ways. AD* [31] is effective for the dynamic and complex shortest path problem. Within allowed computing time, the AD* reuses the previous search efforts and continuously improves the solution. These algorithms iteratively determine the shortest paths using experience of the previous iteration when the arc costs of a graph change.

Generally, learning effects change the shortest path of the graph, i.e., the shortest path of an SPP is distinct from that of an SPLE. For example, there are two paths from s to γ in graph G in Figure 1. One path is $s \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow \gamma$ with total cost 40 and the other contains only arc $s \rightarrow \gamma$ with a total cost 38. Obviously, the second is shorter. However, costs of one arc are different when it is located at different positions when we take learning effects into account. For an example, when we consider learning effects, the normal cost $c(n_i, n_j)$ of arc (n_i, n_j) becomes $c(n_i, n_j, r) = c(n_i, n_j) \times r^{-0.2}$ if arc (n_i, n_j) is located at the r^{th} position of a path. If arcs are traversed in the above order, the arc costs in G are shortened as shown in Figure 2. The total cost of the path $s \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow \gamma$ becomes $10 + 8.71 + 8.03 + 7.58 = 34.31$, which is less than the total cost 38 on the path $s \rightarrow \gamma$, i.e., the shortest path in Figure 2 is different from that in Figure 1. Therefore, A* algorithms and the Dijkstra algorithm for SSPPs are not suitable for the SPLE problem under study. In addition, few of the above properties (e.g., the FIFO) in irregular DSSPs are satisfied in the SPLE, and existing algorithms for DSSPs are not suitable for the SPLE either. In this paper, the new characteristics in SPLE motivated us to develop the AA* (Adapted A*) algorithm for the SPLE. Admissibility, monotonicity, and consistency of AA* are analyzed as they are completely different from those of A*.

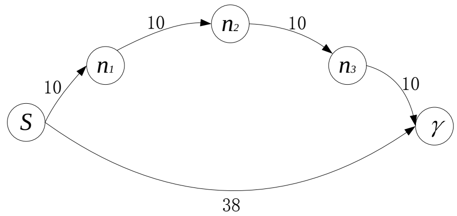

Figure 1: The shortest path in graph G without learning effects

The rest of the paper is organized as follows: The considered problem and some preparations are described in Section 2. Section 3 details the proposed AA* algorithm. Admissibility of the proposal is proven in Section 4. Section 5 redefines consistency (monotonicity) of the proposed algorithm and proves some corresponding properties. Experimental results of compared algorithms are shown in Section 6, followed by conclusions and future research in Section 7.

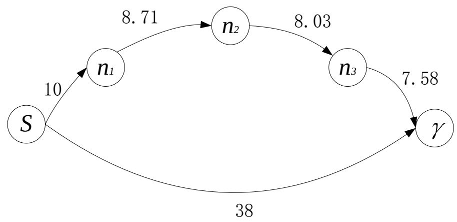

Figure 2: The shortest path in graph G with learning effects

2. Problem Description and Properties

2.1. Problem description

Let G be a finite directed graph $G =< N, A, c >$ with a set of $|N|$ nodes and a set of $|A|$ arcs. Arc $(n, n^{'})$ is labeled with a positive cost $c(n, n^{'}) \in R^{+}$. A path P is a sequence of nodes going from the start node n_0 to some other node n_k in N, i.e., $n_0 \rightarrow n_{[1]} \rightarrow n_{[2]} \rightarrow \ldots \rightarrow n_{[j]} \rightarrow n_k$. $n_{[i]} \in N$ is the i^{th} node on P and $(n_{[i]}, n_{[i+1]}) \in A$ for all $0 < i < j$, $(n_0, n_{[1]}) \in A$, and $(n_{[j]}, n_k) \in A$. For simplicity, let $Q = \{n_{[i]} | 1 \leq i \leq j\}$ and $\pi^Q = (\pi^Q_{[1]}, \pi^Q_{[2]}, \cdots, \pi^Q_{[j]})$ be a permutation of the elements in Q, i.e., $\pi^Q_{[i]} = n_i$. The path is denoted as $P_{(n_0, \pi(Q), n_k)}$. Since an arc can be in several paths with different positions in the shortest path problem with learning effects (SPLE), the cost of the arc (n_i, n_j) is $c(n_i, n_j, r) = c(n_i, n_j) \times r^{\alpha}$ ($r = 1, 2, \ldots, \rho$ and $\rho = \min\{|N| - 1, |A|\}$) if arc (n_i, n_j) is located at the r^{th} position of P, where $\alpha < 0$ is the learning index. The cost $g_l(P)$ of the path P is the sum of the costs of its arcs with learning effects, i.e., $g_l(P) = \sum_{i=0}^{k-1} c(n_{[i]}, n_{[i+1]}, i + 1)$. For a given set of goal nodes $\Gamma \subseteq N$, SPLE tries to find the shortest path P^* with the minimum cost $c(P^*)$ from n_0 to at least one node in Γ.

A* algorithms are commonly used in pathfinding and graph traversals. They are usually best-first search algorithms. As A* traverses the graph, it follows a path with the lowest expected total cost. The cost function $f(n)$ is the sum of two functions, i.e., $f(n) = g(n) + h(n)$, where the past path-cost function $g(n)$ denotes the cost of the path from the start node to the current node n, and the heuristic function $h(n)$ estimates the cost of a path from node n to the goal node(s). Let $h^*(n)$ be the real cost of a path from node n to the goal node(s). The A* algorithm is admissible if $h(n) \leq h^*(n)$ $(\forall n)$, which guarantees that the optimal solution can be found if it exists.

For the problem under study, we consider the following assumptions:

(i) The cost of each arc is greater than some positive number ϵ.

(ii) For any node n in the search graph, $h(n) \leq h^*(n)$, i.e., $h(n)$ does not overestimate the real cost $h^*(n)$ in the graph without learning effects.

Notations used in the following are shown in Table 1.

2.2. Properties

Definition 1. *A vector* $\vec{u}$=$(u_1, u_2, \cdots, u_k)$ *dominates* $\vec{v} = (v_1, v_2, \cdots, v_k)$ *(denoted by* $\vec{u} \prec \vec{v}$ *) iff* $\vec{u}$ *is partially less than* $\vec{v}$*, i.e.,* $\forall i \in$*{1, 2,* $\cdots$*,* k*},* $u_i \leq v_i \bigwedge \exists i \in$ *{1, 2,* $\cdots$*,* k*},* $u_i < v_i$.

Table 1: Notations

$c(n_i, n_j, r)$	Cost of arc (n_i, n_j) when it is located at the r^{th} position in a path. $r = 1, 2, \ldots, \rho$. $\rho = \min\{\lvert N\rvert - 1, \lvert A\rvert\}$).
$g^*(n_i)$	Cheapest cost among the paths from node n_0 to node n_i in G without learning effects.
$h^*(n_i)$	Cheapest cost among the paths from node n_i to Γ in G without learning effects.
$h(n_i)$	Estimated cost on $h^*(n_i)$.
$g_l(n_i, r)$	Cost of the path going from node n_0 to node n_i which contains r arcs of G with learning effect.
$h_l^*(n_i, r)$	Cheapest cost of paths going from node n_i to Γ in N with learning effect where node n_i is located at the r^{th} position.
$h_l(n_i)$	Estimated cost of $h_l^*(n_i, r)$ ($r \in [1, \rho]$, $h_l(n_i) = \rho^\alpha \times h(n_i)$.
$g_l(P)$	Total cost of arcs on path P in G with learning effects.
$f_l(n_i, r)$	Estimated cost of the path going from node n_0 to Γ through node n_i in G with learning effects, i.e., $f_l(n_i, r) = g_l(n_i, r) + h_l(n_i)$.
$P_{(n_i, \pi(Q), n_j)}$	Path starting from node n_i to the sink node n_j through the node sequence $\pi(Q)$ ($Q \subset N - \{n_i, n_j\}$).
$S^P_{(n_{[i]}, n_{[j]})}$	Sub-path of P from node $n_{[i]}$ to node $n_{[j]}$.
SG	Acyclic search graph storing promising partial solution paths.
$\vec{g}$	$\vec{g} = (g_l(n_i, r), \rho - r)$ denotes the path containing r arcs with cost $g_l(n_i, r)$.
$G_{op}(n_i)$	Set of paths reaching node n_i whose extending nodes have not been explored. i.e., of which each element is a vector $\vec{g}$.
$G_{cl}(n_i)$	Set of paths reaching node n_i whose extending nodes have been explored. i.e., of which each element is a vector $\vec{g}$.
$OPEN$	List of partial solution paths that can be further expanded, of which each element is a tuple $(n_i, \vec{g}, f_l(n_i, r))$. $OPEN$ is stored in a heap structure for a fast selection of the path with minimum f_l.
c^*	Cheapest cost of the path going from node n_0 to Γ in G with learning effect, i.e., $c^* = h_l^*(n_0)$.
C	Cheapest cost of the path found so far from node n_0 to Γ
$goal$	Goal nodes with the cheapest found cost so far.

Let $P_{(n_0, \pi(Q_1), n_i)}$ and $P_{(n_0, \pi(Q_2), n_i)}$ be two sub-paths from the start node n_0 to node n_i ($n_i \in N$) with costs $\vec{g_1} = (g_l^1, \rho - r_1)$ and $\vec{g_2} = (g_l^2, \rho - r_2)$ respectively. Two paths P_1 and P_2 are constructed by combining $P_{(n_0, \pi(Q_1), n_i)}$ and $P_{(n_0, \pi(Q_2), n_i)}$ with another sub-path $P_{(n_i, \pi(Q_3), \gamma)}$ where Q_3 is a subset of $N - \Gamma - \{n_0, n_i\} - Q_1 - Q_2$ and $\gamma \in \Gamma$.

Theorem 1. *If $\vec{g_1} \prec \vec{g_2}$, then $g_l(P_1) < g_l(P_2)$.*

Proof.

$$\begin{aligned} g_l(P_1) =& g_l^1 + (r_1 + 1)^\alpha c(n_i, \pi^{Q_3}_{[1]}) + \sum_{i=2}^{|Q_3|} (r_1 + i)^\alpha c(\pi^{Q_3}_{[i-1]}, \pi^{Q_3}_{[i]}) \\ &+ (r_1 + |Q_3| + 1)^\alpha c(\pi^{Q_3}_{[|Q_3|]}, \gamma) \end{aligned}$$

$$g_l(P_2) = g_l^1 + (r_2+1)^\alpha c(n_i, \pi_{[1]}^{Q_3}) + \sum_{i=2}^{|Q_3|} (r_2+i)^\alpha c(\pi_{[i-1]}^{Q_3}, \pi_{[i]}^{Q_3})$$
$$+ (r_2 + |Q_3| + 1)^\alpha c(\pi_{[|Q_3|]}^{Q_3}, \gamma)$$

Then,

$$g_l(P_2) - g_l(P_1) = (g_l^2 - g_l^1) + ((r_2+1)^\alpha - (r_1+1)^\alpha) c(n_i, \pi_{[1]}^{Q_3})$$
$$+ \sum_{i=2}^{|Q_3|} (r_2+i)((r_2+i)^\alpha - (r_1+i)^\alpha) c(\pi_{[i-1]}^{Q_3}, \pi_{[i]}^{Q_3})$$
$$+ ((r_2 + |Q_3| + 1)^\alpha - (r_1 + |Q_3| + 1)^\alpha) c(\pi_{[|Q_3|]}^{Q_3}, \gamma)$$

$\vec{g_1} \prec \vec{g_2}$ implies that (i) $g_l^1 < g_l^2, \rho - r_1 \le \rho - r_2$ or (ii) $g_l^1 = g_l^2, \rho - r_1 < \rho - r_2$.

For the case (i), $\rho - r_1 \le \rho - r_2$ means that $r_1 \ge r_2$. Similarly, $f(x) = x^\alpha$ is a decreasing function since $\alpha < 0$. Therefore $(r_2+i)^\alpha - (r_1+i)^\alpha \ge 0,\ \forall i \in \{1, \cdots, |Q_3|+1\}$. According to $g_l^1 < g_l^2$, we obtain that $g_l(P_2) - g_l(P_1) > 0$, i.e., $g_l(P_2) > g_l(P_1)$.

For the case (ii), $g_l^1 = g_l^2$ and $r_1 > r_2$. Because $f(x) = x^\alpha$ ($\alpha < 0$) is a decreasing function, $(r_2+i)^\alpha - (r_1+i)^\alpha > 0,\ \forall i \in \{1, \cdots, |Q_3|+1\}$. In addition, $g_l^1 = g_l^2$, so $g_l(P_2) > g_l(P_1)$.

The proof completes both cases of $\vec{g_1} \prec \vec{g_2}$. □

Theorem 2. *If $\vec{g_1} \prec \vec{g_2}$, then $\vec{g}(P_1) \prec \vec{g}(P_2)$.*

Proof. Because sub-path $P_{(n_i, \pi(Q_3), \gamma)}$ has $|Q_3|+1$ arcs, there are $r_1 + |Q_3| + 1$ and $r_2 + |Q_3| + 1$ arcs on P_1 and P_2 respectively. $\vec{g}(P_1) = (g_1(P_1), \rho - (r_1 + |Q_3| + 1))$, $\vec{g}(P_2) = (g_1(P_2), \rho - (r_2 + |Q_3| + 1))$. $\vec{g_1} \prec \vec{g_2}$ implies that $\rho - (r_1 + |Q_3| + 1) \le \rho - (r_2 + |Q_3| + 1)$ and $g_l(P_1) < g_l(P_2)$. Therefore $\vec{g}(P_1) \prec \vec{g}(P_2)$. □

For the shortest path $P_{(n_0, \pi(Q), \gamma)}$ ($\gamma \in \Gamma$ and $Q \subseteq N - n_0 - \Gamma$) in graph G without learning effects, every sub-path $S_{(n_0, n_i)}^{P_{(n_0, \pi(Q), \gamma)}}$ ($n_i \in Q$) is the shortest path from the start node n_0 to node n_i. That is to say, global optimality implies local optimality. However, this is not true in the SPLE problem studied in this paper.

Theorem 3. *Let $P_{(n_0, \pi(Q), \gamma)}(Q \subseteq N - \{n_0\} - \Gamma,\ \gamma \in \Gamma)$ be one of the shortest paths from start node n_0 to Γ in graph G with learning effects. The shortest path from node n_0 to node n_i ($n_i \in Q$) in G with learning effects might not be a sub-path of $P_{(n_0, \pi(Q), \gamma)}$.*

Proof. Let $P_{(n_0, \pi(Q_1), n_i)}(Q_1 \subseteq N - \{n_0, n_i\} - \Gamma)$ be the shortest path from n_0 to node n_i with cost $\vec{g_1} = (g_l^1, r_1)$ in G with learning effects. Assume there exists another path $P_{(n_0, \pi(Q_2), n_i)}(Q_2 \subseteq N - \{n_0, n_i\} - \Gamma)$ with cost $\vec{g_2} = (g_l^2, r_2)$ and $g_l^1 < g_l^2$. Two paths P_1 and P_2 are constructed by combining $P_{(n_0, \pi(Q_1), n_i)}$ and $P_{(n_0, \pi(Q_2), n_i)}$ with sub-path $P_{(n_i, \pi(Q_3), \gamma)}$ where $Q_3 \subseteq N - \Gamma - \{n_0, n_i\} - Q_1 - Q_2$. There are two cases:

(i) $r_1 \ge r_2$. Conditions $g_l^1 < g_l^2$ and $r_1 \ge r_2$ imply that $\vec{g_1} \prec \vec{g_2}$. According to Theorem 1, P_1 is shorter than P_2.

(ii) $r_1 < r_2$. However, it is uncertain which one among P_1 and P_2 is better when $g_l^1 < g_l^2$ and $r_1 < r_2$, especially when g_l^1 is slightly less than g_l^2 and r_2 is much greater than r_1. Because of learning effects and $r_1 < r_2$, $g_l(P_2) - g_l^2$, the cost of $S_{(n_i,\gamma)}^{P_{(n_0,\pi(Q),\gamma)}}$ on P_2, is less than $g_l(P_1) - g_l^1$. Therefore, it is possible that $[g_l(P_1) - g_l^1] - [g_l(P_2) - g_l^2] > g_l^2 - g_l^1$, i.e., P_1 is not $P_{(n_0,\pi(Q),\gamma)}$ though P_1 contains the shortest sub-path $P_{(n_0,\pi(Q_1),n_i)}$. □

To illustrate the case (ii) of Theorem 3, an example is now given. Assume node n_i is the previous node of γ on the shortest path and $\vec{g}(P_{(n_0,\pi(Q_1),n_i)}) = (100, 99)$, $\vec{g}(P_{(n_0,\pi(Q_2),n_i)}) = (95, 1)$, $c(n_i, \gamma) = 20$. If $\alpha = -0.2$, then $c(n_i, \gamma, 100) = 100^{-0.2} \times 20 = 7.96$ and $c(n_i, \gamma, 2) = 2^{-0.2} \times 20 = 17.41$. Therefore $g_l(P_1) = 100 + 7.96 = 107.96$ and $g_l(P_2) = 95 + 17.41 = 112.41$, i.e., P_1 is shorter than P_2.

Theorem 4. *Let $P_{(n_0,\pi(Q),\gamma)}(Q \subseteq N - \{n_0\} - \Gamma,\ \gamma \in \Gamma)$ be one of the shortest paths from the start node n_0 to Γ in graph G with learning effects. Every sub-path $S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),\gamma)}}$ $(n_i \in Q)$ is one of the nondominated paths from the start node n_0 to node n_i.*

Proof. Assume that sub-path $S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),\gamma)}}$ is not a non-dominated path from the start node n_0 to node n_i with cost $\vec{g}$. There must exist a path $P_{(n_0,\pi(Q_1),n_i)}(Q_1 \subseteq N - \{n_0, n_i\} - \Gamma)$ with cost $\vec{g}'$ and $\vec{g}' \prec \vec{g}$. A new path $P_{(n_0,\pi(Q'),\gamma)}$ $(Q' \subseteq N - \{n_0\} - \Gamma)$ can be generated by combining $P_{(n_0,\pi(Q_1),n_i)}$ with the other sub-path $S_{(n_i,\gamma)}^{P_{(n_0,\pi(Q),\gamma)}}$. According to Theorem 1, $g_l(P_{(n_0,\pi(Q'),\gamma)}) < g_l(P_{(n_0,\pi(Q),\gamma)})$, which is a contradiction to the optimality of path $P_{(n_0,\pi(Q),\gamma)}$. Therefore, sub-path $S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),\gamma)}}$ is one of the nondominated paths from the start node n_0 to node n_i. □

3. Adapted A* for SPLE problems

Theorems 3 and 4 illustrate that not only the shortest path P from the start node n_0 to some node n_i $(n_i \in N)$ but also some other sub-paths need to be put in the list of solution paths to be explored ($OPEN$). If the path from n_0 to n_i contains more arcs, more learning effects are accumulated and then the total cost from n_i to Γ might decrease. Therefore, both the cost g_l and the included number of arcs r on the paths are included in the graph SG. For the paths from the same start node to the same sink node and with the same cost g_l, the bigger the r (or equivalently the smaller the $\rho - r$) implies a higher "learning effect" and a reduced total cost of the remaining sub-path.

Adapted A* (AA* for short) for the SPLE is a best-first heuristic search algorithm, adapted from A*. A seed solution path in G without learning effects $n_0 \to n_{[1]} \to n_{[2]} \to \cdots \to n_{[k]} \to \gamma$ is generated by the weighted A* [32]. And we obtain $C = c(n_0, n_{[1]}) + 2^{\alpha} c(n_{[1]}, n_{[2]}) + \cdots + (k+1)^{\alpha} c(n_{[k]}, \gamma)$ as an upper bound. AA* starts the search process with the start node n_0. Initially, n_0 is set as the only node in the search graph SG and the sub-path tuple $(n_0,\ \vec{g},\ f_l(n_0, 0))$ is introduced into the list $OPEN$. $OPEN$ stores all the alternatives to be expanded, which are stored using a heap structure for a quick selection and retrieval. In every iteration, AA* selects the path P with the smallest f_l (randomly select one to break ties if there are any), which is determined by $g_l + h_l$. Each successor of P is expanded by deleting P from $OPEN$ and moving the corresponding $\vec{g}$ from G_{op} to G_{cl}. A* uses a search tree to record the shortest paths from the start node to the expanded nodes, e.g., only the shortest path from the source node to node n_i is recorded in the tree if more than one path from the source node to the same node n_i has been found. However, AA* uses the search graph SG to store all the non-dominated paths found to the same node n_i in terms of Theorem 4. $G_{op}(n_i)$ and $G_{cl}(n_i)$ are two path sets with n_i being the sink node. $G_{op}(n_i)$ contains the expanded paths and $G_{cl}(n_i)$ contains the unexpended ones respectively.

Let P' be a new path to node n_i which is constructed during the expansion with an estimated cost $f_l(P')$ being calculated by $g_l(P') + h_l(P')$. If $n_i \in \Gamma$ and $f_l(P') < C$, C is updated to $g_l(P')$ (denoted as $F_{UP} \leftarrow true$). P' and all the paths in $G_{op}(n_i) \bigcup G_{cl}(n_i)$ are verified and three operations are carried out: (i) PRUNE is performed if there are dominations, i.e., P' is discarded if it is dominated by some element $P'' \in G_{op}(n_i) \bigcup G_{cl}(n_i)$, or P'' is removed if P' dominates P'' by deleting its tuples from both $G_{op}(n_i) \bigcup G_{cl}(n_i)$ and $OPEN$ (if the tuple of P'' was already in $OPEN$). (ii) FILTER is performed if there are some bad path(s), i.e., P' is discarded if $f_l(P') > C$ or $P'' \in G_{op}(n_i)$ is removed if $f_l(P'') > C$ when $F_{UP} = true$ by deleting its tuples from both $G_{op}(n_i)$ and $OPEN$. Otherwise, (iii) ENTER is performed by inserting the tuples of P' into both $G_{op}(n_i)$ and $OPEN$. The process is repeated until $OPEN$ is empty, i.e., no path can be expanded. The shortest paths from the start node to the goal nodes are constructed by backtracking from SG.

Let $\vec{g}$ be the cost vector of a path from the start node to the current node m. PRUNE returns $true$ when the new expanded path is discarded. PRUNE is formally described in Algorithm 1.

ALGORITHM 1: Boolean PRUNE($m, \vec{g}$)

begin
  if $m \notin \Gamma$ **then**
    foreach $\vec{g'} \in G_{op}(m) \cup G_{cl}(m)$ **do**
      if $\vec{g'} \prec \vec{g}$ **then**
        return *true*;
      if $\vec{g} \prec \vec{g'}$ **then**
        Eliminate $\vec{g'}$ from $G_{op}(m) \cup G_{cl}(m)$;
        Remove the arc (n, m) ($n \in Predecessor(m)$) from the path with $\vec{g'}$ in SG;
        if $\vec{g'} \in G_{op}(m)$ **then**
          Eliminate $(m, \vec{g'}, f')$ from $OPEN$;
  return *false*.

Let $(m,\ \vec{g},\ f_l)$ be the tuple in $OPEN$ of a path from the start node to the current node m, and c be the cheapest solution cost found so far. FILTER returns $true$ if the new expanded path is discarded. FILTER is formally described in Algorithm 2.

The pseudocode of AA^* for the SPLE is formally described in Algorithm 3.

To illustrate the AA* algorithm, a labeled directed graph is given in Figure 3, where n_0 is the start node and γ is the only goal node. The graph contains 8 nodes and 14 arcs. Because no cycle is included on the path from the start node n_0 to Γ, there are at most $\rho = min\{8-1, 14\} = 7$ arcs. The learning effect factor α takes a value of -0.2. C is initialized as $+\infty$. $c(n, \gamma)$ is the cost of arc

ALGORITHM 2: Boolean FILTER($m, n, \vec{g}, \ f_l, F_{UP}$)

begin
  if $f_l > C$ **then**
    return *true*;
  if $f_l = C$ ***and*** $F_{UP} = false$ **then**
    return *false*;
  if $F_{UP} = true$ **then**
    foreach $(m, \vec{g'}, f'_l) \in OPEN$ **do**
      if $f'_l > C$ **then**
        Delete the tuple $(m, \vec{g'}, f'_l)$ in $OPEN$ and $\vec{g'}$ in $G_{op}(m)$;
        Remove the arc (n, m) ($n \in Predecessor(m)$) from the path with the estimate f' in SG;
  return *false*.

(n, γ) if it exists in G, otherwise it is $+\infty$. In the graph G without learning effects, the heuristic value h is defined as $h(n) = \min\{c(n, \gamma), \min_{n_s \in Sucessor(n)} c(n, n_s) + \min_{n_p \in Predecessor(\gamma)} c(n_p, \gamma)\}$. In the graph G with learning effects, the heuristic value $h_l(n) = \rho^\alpha \times h$ is used to evaluate the value of h_l^* (h_l is computed by $\rho^\alpha \times h = 7^{-0.2} \times h$ in this example). h and h_l for each node are given in Table 2.

Figures 4-9 illustrate the changes on the search graph when AA* is performed on the example, which show the operations PRUNE, FILTER, and the updating operations on G_{op}, G_{cl} sets and $OPEN$. A trace of $OPEN$ is seen in Table 3. Details for the data-structures in each iteration are given in the following.

(1) n_0 is initialized as the root and the only node of the search graph SG. Therefore $g_l(n_0, 0) = 0$, $\rho{-}0 = 7$, $G_{op}(n_0) \leftarrow \{(0, 7)\}$, and $G_{cl}(n_0) = \varnothing$. $f_l = g_l(n_0, 0) + h_l(n_0) = 0 + 2.033 = 2.033$. Therefore $OPEN \leftarrow \{(n_0, (0, 7), 2.033)\}$.

(2) The only path in $OPEN$ is selected, of which the four extensions n_1, n_2, n_3, n_4 are added to SG and $OPEN$. The corresponding four arcs are located at the first positions of the generated search paths respectively. The learning effect has no impact on their costs, i.e., they are unchanged. Since $g_l(n_1, 1) = 6$ and there are at most 6 arcs in any path from node n_1 to goal node γ, $\vec{g}(n_1, 1) = (6, 6)$. $f_l(n_1, 1) = g_l(n_1, 1) + h_l(n_1) = 6 + 3.338 = 9.338$. $G_{op}(n_1) \leftarrow \{(6, 6)\}$ and $(n_1, (6, 6), 9.338)$ is inserted into $OPEN$. The other extensions are processed in the same way. The resulting SG is depicted in Figure 4.

(3) Node n_3, with the smallest estimated cost in $OPEN$, is selected for extension. n_1 is the only offspring of n_3. The arc (n_3, n_1) is located at the 2^{nd} position in a new path. Because of the learning effect, the cost of the arc $c(n_3, n_1, 2)$ is $2 \times 2^{-0.2} = 1.741$. The new sub-path to node n_1 dominates the existing ones in $OPEN$. According to Theorem 1, a path to γ including

ALGORITHM 3: Algorithm AA* for SPLE

Input: A finite labeled directed graph $G = (N, A, c)$, a start node $n_0 \in N$, a set of goal nodes $\Gamma \subseteq N$, a constant learning index $\alpha \leq 0$

Output: The minimum cost paths in G from n_0 to Γ

begin
- $C \leftarrow +\infty$;
- Set n_0 as the root of the acyclic search graph SG;
- $\rho \leftarrow min\{|N| - 1, |A|\}$, $g_l(n_0, 0) \leftarrow 0$, $\vec{g}(n_0, 0) \leftarrow (g_l(n_0, 0), \rho)$, $h_l(n_0) \leftarrow \rho^\alpha \times h(n_0)$, $f_l(n_0, 0) \leftarrow g_l(n_0, 0) + h_l(n_0)$;
- $G_{op}(n_0) \leftarrow \{\vec{g}(n_0, 0)\}$, $G_{cl} \leftarrow \varnothing$, $OPEN \leftarrow \{(n_0, \vec{g}(n_0, 0), f_l(n_0, 0))\}$;
- **while** $OPEN \neq \varnothing$ **do**
  - Select the path $(n, \vec{g}(n, r), f_l(n, r))$ in $OPEN$ with the lowest f_l;
  - Delete $(n, \vec{g}(n, r), f_l(n, r))$ from $OPEN$;
  - Move $\vec{g}(n, r)$ from $G_{op}(n)$ to $G_{cl}(n)$;
  - **if** $n \notin \Gamma$ **then**
    - Generate the set M by **expanding** node n, which contains only the successors not already ancestors of node n in SG ;
    - **foreach** $m \in M$ **do**
      - $F_{UP} \leftarrow false$, $g_l(m, r + 1) \leftarrow g_l(n, r) + c(n, m) \times (r + 1)^\alpha$;
      - $\vec{g}(m, r + 1) \leftarrow (g_l(m, r + 1), \rho - (r + 1))$, $f_l(m, r + 1) \leftarrow g_l(m, r + 1) + \rho^\alpha \times h(m)$;
      - **if** $m \in \Gamma \&\& f_l(m, r + 1) < C$ **then**
        - $C \leftarrow g_l(m, r + 1)$, $goal \leftarrow \{m\}$, $F_{UP} \leftarrow true$;
      - **if** $PRUNE(m, \vec{g}) = false$ **then**
        - **if** $FILTER(m, f_l) = false$ **then**
          - Establish a pointer from m to n with the cost $\vec{g}(m, r + 1)$ in SG;
          - Insert $(m, \vec{g}(m, r + 1), f_l(m, r + 1))$ to $OPEN$;
          - $G_{op}(m) \leftarrow G_{op}(m) \bigcup \{\vec{g}(m, r + 1)\}$;
          - /* A path from n_0 to a new goal node in Γ with cost $g_l = C$ is found. */
          - **if** $F_{UP} = false$ *and* $g_l = C$ *and* $m \in \Gamma$ *and* $m \notin goal$ **then**
            - $goal \leftarrow goal \bigcup \{m\}$;
- Construct the subgraph of SG by backtracking the nodes in *goal* according to the cost C;
- **return** *The paths from goal to the start node.*

the new sub-path is shorter than that of the one including the existing sub-path. By PRUNE, the arc (n_1, n_0) is removed from SG, the tuple $(n_1, (6, 6), 9.338)$ is deleted from $OPEN$, and $\vec{g}(n_1, 1)$ is eliminated from $G_{op}(n_1)$. Similarly, the arc (n_1, n_3) is added to SG. The tuple $(n_1, (2.741, 5), 6.129)$ is inserted into $OPEN$, and the vector $\vec{g}(n_1, 2) = (2.741, 5)$ is appended to $G_{op}(n_1)$. The n_3 expanding process is shown in Figure 5.

(4) Node n_1 is selected for expansion from $OPEN$ because it had the smallest cost estimation. There are two direct successors, n_4 and n_6. The cost $c(n_1, n_4, 3)$ is $3 \times 3^{-0.2} = 2.408$ because the arc is located at the 3^{rd} position in the path. Therefore, $g_l(n_4, 3) = g_l(n_1, 2) + c(n_1, n_4, 3) = 2.741 + 2.408 = 5.149$ and $\vec{g}(n_4, 3) = (5.149, 4)$. Now there are two paths reaching n_4, of which the cost vectors do not dominate each other. In terms of Theorems 3

Table 2: Heuristic Function

n	n_0	n_1	n_2	n_3	n_4	n_5	n_6	γ
$h(n)$	3	5	2	4	5	3	5	0
$h_l(n) = 7^{-0.2} \times h(n)$	2.033	3.338	1.355	2.710	3.388	2.033	3.388	0

and 4, both sub-paths are stored into SG, i.e., the arc (n_4, n_1) is added to SG. The tuple $(n_4, (5.149, 4), 8.537)$ is inserted into $OPEN$, and the vector $\vec{g}(n_4, 3) = (5.149, 4)$ is appended to $G_{op}(n_4)$. For the node n_6, $c(n_1, n_6, 3) = 6 \times 3^{-0.2} = 4.816$, $g_l(n_6, 3) = 7.558$, $f_l(n_6, 3) = 10.945$. The tuple $(n_6, (7.558, 4), 10.945)$ is inserted into $OPEN$, and the vector $\vec{g}(n_6, 3) = (7.558, 4)$ is appended to $G_{op}(n_6)$. The n_1 expanding process is shown in Figure 6.

(5) Node n_4 is selected because its f_l is the cheapest in $OPEN$, from which n_6 and γ are extended. The cost of the new path leading to the goal node γ is 9.353, which is cheaper than C. Therefore C is updated to 9.353, and the path to n_2 is filtered because $f_l(n_2) = 9.355 > C$ using steps 8 and 9 in FILTER. And the path to n_6 is filtered since $f_l(n_6) = 10.945 > C$. A pointer from γ to n_4 is added to SG. The tuple $(\gamma, (9.353, 5), 9.353)$ is inserted into $OPEN$, and the vector $\vec{g}(\gamma, 2) = (9.353, 5)$ is appended to $G_{op}(\gamma)$. As for n_6, the cost of arc (n_4, n_6) is now $c(n_4, n_6, 2) = 4 \times 2^{-0.2} = 3.482$. $g_l(n_6, 2) = 5 + 3.482 = 8.482$, and $f_l(n_6, 2) = g_l(n_6, 2) + h_l(n_6) = 8.482 + 3.388 = 12.860$. Therefore, the path to n_6 is discarded by FILTER because $f_l(n_6, 2) > C$. The n_4 expanding process is shown in Figure 7.

(6) Next, the second path to n_4 (with $\vec{g}(n_4, 3) = (5.149, 4)$) is selected, as it has the smallest f_l in $OPEN$. The two direct successors γ and n_6 of n_4 are checked again. The arc (n_4, γ) is located at the 4^{th} position in the newly generated path with the cost $c(n_4, \gamma, 4) = 5 \times 4^{-0.2} = 3.789$. Therefore, $f_l(\gamma, 4) = g_l(\gamma, 4) = g_l(n_4, 3) + c(n_4, \gamma, 4) = 5.149 + 3.789 = 8.938$ which is less than $C = 9.353$, and C is updated to 8.938. The extension to γ is generated. The cost of arc (n_4, n_6) is changed to $c(n_4, n_6, 4) = 4 \times 4^{-0.2} = 3.031$. $g_l(n_6, 4) = 8.180$, $f_l(n_6, 4) = g_l(n_6, 4) + h_l(n_6) = 8.180 + 3.388 = 11.568 > C = 8.938$. Therefore, the sub-path to n_6 is discarded by FILTER. The resulting SG is depicted in Figure 8.

(7) Now the only remaining alternative $(\gamma, (8.938, 3), 8.938)$ in $OPEN$ is selected and removed from $OPEN$. $OPEN$ is now empty. The algorithm traces back the obtained SG from γ. The obtained path with cost 8.938 is returned, as demonstrated in Figure 9.

AA* records the non-dominated sub-paths in the search graph. Though it is similar to NAMOA* (New Approach to Multi-Objective A*)[20], there are several differences between them: (i) AA* is for a single objective while NAMOA* was for multiple objectives, i.e., AA* returns the shortest solution paths while NAMOA* gives the optimal Pareto solution set. (ii) Along one optimal solution path, all the elements of $\vec{g}$ increase simultaneously in NAMOA* whereas it is not the case in AA*. In the cost vector of any path $\overrightarrow{g} = (g_l(n_i, r), \rho - r)$ of the AA*, the first element $g_l(n_i, r)$ increases along the solution path while the second one $\rho - r$ decreases. However, there are multiple elements in the cost vector $\overrightarrow{g}$ of NAMOA* and all the elements increase simultaneously.

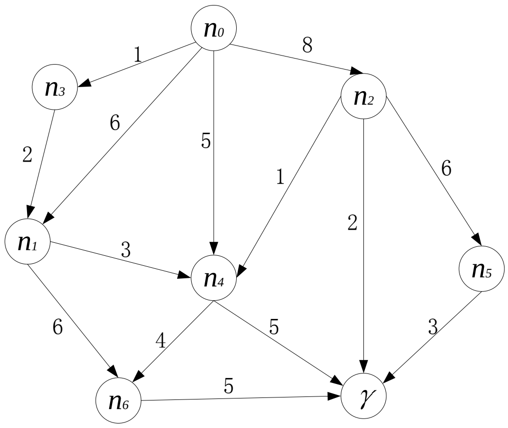

Figure 3: Sample graph

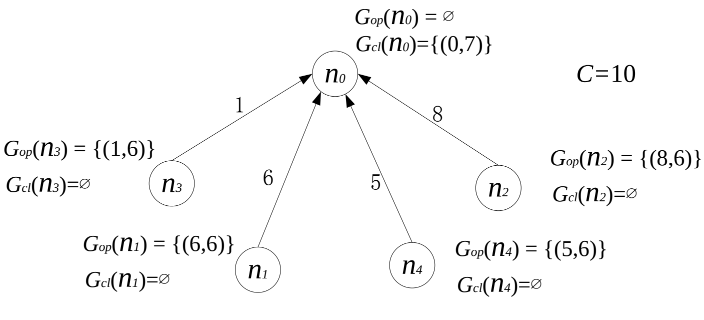

Figure 4: Search graph (iteration 2)

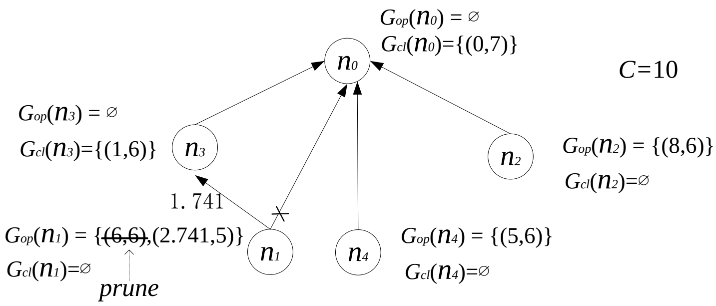

Figure 5: Search graph (iteration 3)

(iii) NAMOA* considers the static shortest path problems but AA* deals with the regular dynamic shortest path version. These distinct aspects lead to different properties, which are analyzed below.

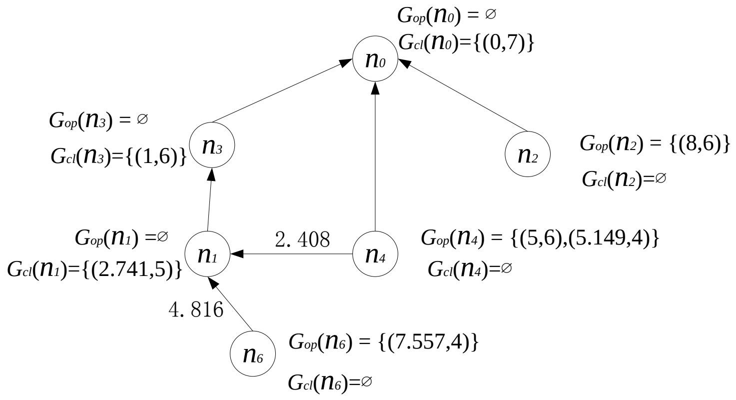

Figure 6: Search graph (iteration 4)

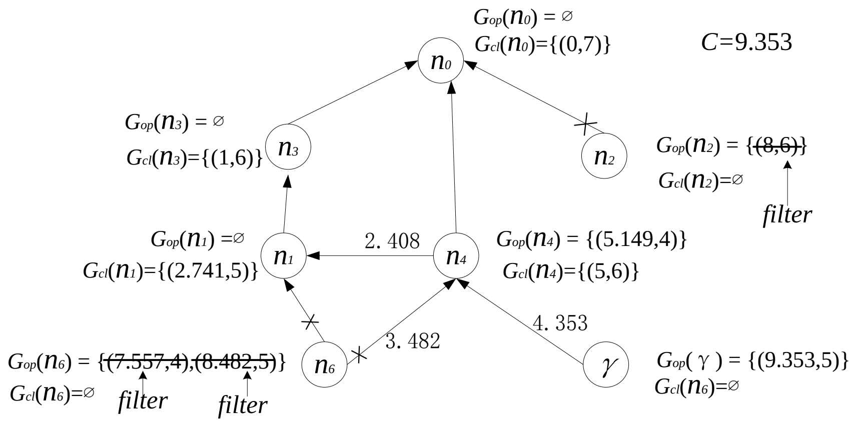

Figure 7: Search graph (iteration 5)

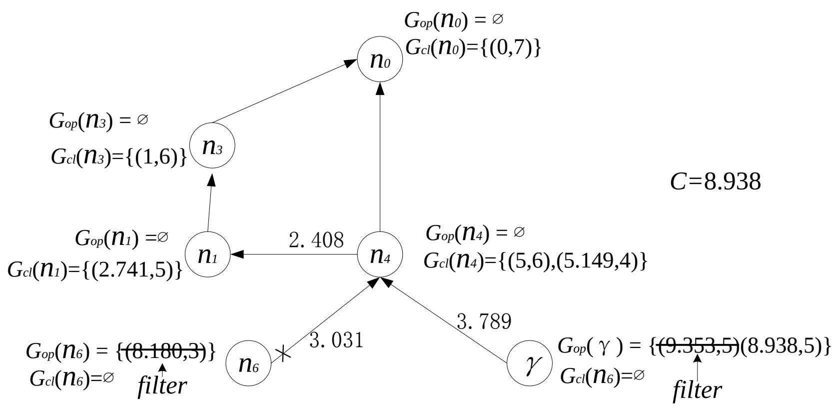

Figure 8: Search graph (iteration 6)

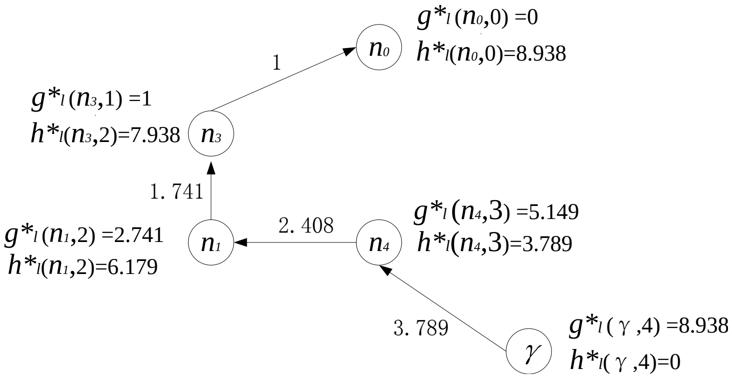

Figure 9: The final solution subgraph

4. Admissibility of AA*

An algorithm is *admissible* if it is guaranteed to return an optimal solution whenever a solution exists [33]. Let $h_l^*(n_i, r)$ be the cheapest cost of the paths going from node n_i to Γ in G with learning effects where node n_i is located at the r^{th} position. Similar to the definition of admissible heuristics [33] for A*, we define admissible heuristics for AA*:

Definition 2. *A heuristic function h_l is admissible if $h_l(n) \leq h_l^*(n, r)$ ($r = 0, 1, 2, \cdots, \rho$) for each node $n \in G$.*

Theorem 5. *If $h(n)$ is admissible in the graph G without learning effects, then $h_l(n) = \rho^\alpha h(n)$ is admissible in G with learning effects.*

Proof. Let $P_{(n_i, \pi(Q), \gamma)}$ ($Q \subset N - \Gamma - \{n_i\}$ and $\gamma \in \Gamma$) be the shortest path from node n_i to Γ and n_i located at the r^{th} position in G with learning effects. Then $\forall r(r + |Q| + 1 \leq \rho)$:

$$\begin{aligned} h_l^*(n_i, r) &= c(n_i, \pi_{[1]}^Q, r+1) + \textstyle\sum_{j=1}^{|Q|-1} c(\pi_{[j]}^Q, \pi_{[j+1]}^Q, r+j+1) + \\ &\quad c(\pi_{[|Q|]}^Q, \gamma, r + |Q| + 1) \\ &= (r+1)^\alpha c(n_i, \pi_{[1]}^Q) + \textstyle\sum_{j=1}^{|Q|-1} (r+j+1)^\alpha c(\pi_{[j]}^Q, \pi_{[j+1]}^Q) + \\ &\quad (r + |Q| + 1)^\alpha c(\pi_{|Q|}^Q, \gamma) \end{aligned}$$

Since $f(x) = x^\alpha$ ($\alpha < 0$) is a decreasing function and $r + |Q| + 1 \leq \rho$, we obtain:

$$\begin{aligned} h_l^*(n_i, r) &\geq \rho^\alpha c(n_i, \pi_{[1]}^Q) + \textstyle\sum_{j=1}^{|Q|-1} \rho^\alpha c(\pi_{[j]}^Q, \pi_{[j+1]}^Q) + \rho^\alpha c(\pi_{|Q|}^Q, \gamma) \\ &= \rho^\alpha [c(n_i, \pi_{[1]}^Q) + \textstyle\sum_{j=1}^{|Q|-1} c(\pi_{[j]}^Q, \pi_{[j+1]}^Q) + c(\pi_{|Q|}^Q, \gamma)] \end{aligned}$$

Because $h^*(n_i)$ is the cheapest cost of the path from n_i to Γ in the graph G without learning effects, then $c(n_i, \pi_{[1]}^Q) + \sum_{j=1}^{|Q|-1} c(\pi_{[j]}^Q, \pi_{[j+1]}^Q) + c(\pi_{|Q|}^Q, \gamma) \geq h^*(n_i)$.

Therefore, $h_l^*(n_i, r) \geq \rho^\alpha h^*(n_i)$.

If $h(n)$ is admissible, it implies that $h^*(n_i) \geq h(n_i)$. Therefore, $h_l(n_i) = \rho^\alpha \times h(n_i) \leq \rho^\alpha \times h^*(n_i) \leq h_l^*(n_i, r)$, i.e., $h_l(n) = \rho^\alpha h(n)$ is admissible in G with learning effects. □

Table 3: Tuples in OPEN at each iteration of AA^*

Iteration	OPEN
1	$(n_0, (0,7), 2.033) \leftarrow$
2	$(n_1, (6,6), 9.388)$
	$(n_2, (8,6), 9.355)$
	$(n_3, (1,6), 3.710) \leftarrow$
	$(n_4, (5,6), 8.388)$
3	$(n_2, (8,6), 9.355)$
	$(n_4, (5,6), 8.388)$
	$(n_1, (2.741,5), 6.129) \leftarrow$
4	$(n_2, (8,6), 9.355)$
	$(n_4, (5,6), 8.388) \leftarrow$
	$(n_4, (5.149,4), 8.537)$
	$(n_6, (7.558,4), 10.945)$
5	$(n_4, (5.149,4), 8.537) \leftarrow$
	$(\gamma, (9.353,5), 9.353)$
6	$(\gamma, (8.938,3), 8.938) \leftarrow$

Lemma 1. *Let $P_{(n_0,\pi(Q),\gamma)}(Q \subseteq N - \{n_0\} - \Gamma,\ \gamma \in \Gamma)$ be one of the shortest paths from the start node n_0 to Γ in the graph G with learning effects. For any sub-path $S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),\gamma)}}$ ($n_i \in Q$) in $OPEN$, $f_l(S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),\gamma)}}) \leq c^*$.*

Proof. According to Theorem 5, $h_l(n_i) \leq h_l^*(n_i, r)$ $(r = 0, 1, \cdots, \rho)$. Therefore, $f_l(S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),\gamma)}}) = g_l(S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),\gamma)}}) + h_l(n_i) \leq g_l(S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),\gamma)}}) + h_l^*(n_i, r) = c^*$. □

Lemma 2. *For any shortest path from the start node n_0 to Γ, $P_{(n_0,\pi(Q),\gamma)}(Q \subseteq N - \{n_0\} - \Gamma,\ \gamma \in \Gamma)$, in the graph G with learning effects, there is always a sub-path $S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),\gamma)}}$ ($n_i \in Q \cup \{n_0\}$) stores into SG, $G_{op}(n_i)$, and $OPEN$ in each iteration before completing the construction of $P_{(n_0,\pi(Q),\gamma)}$.*

Proof. (Mathematical Induction) **Base case**: At the beginning search of AA^*, only n_0 is selected for expansion. The path including only node n_0 is a sub-path of $P_{(n_0,\pi(Q),\gamma)}$, i.e., $S_{(n_0,n_0)}^{P_{(n_0,\pi(Q),\gamma)}}$ is stored into SG and $\vec{g}(S_{(n_0,n_0)}^{P_{(n_0,\pi(Q),\gamma)}}) \in G_{op}(n_0)$.

Induction step: We assume that the conclusion of the lemma is true in the k^{th} iteration, i.e., there is always a sub-path (for simplicity, it is denoted as S_k^P) of $P_{(n_0,\pi(Q),\gamma)}$ stored into SG, G_{op}, and $OPEN$ in iteration k. Now we would prove it true in iteration $k+1$, i.e., there must be a sub-path S_{k+1}^P in SG, G_{op}, and $OPEN$ in iteration $k+1$. There are two cases for S_k^P in iteration $k+1$.

1. S_k^P is not selected for expansion. Since S_k^P is a sub-path of $P_{(n_0,\pi(Q),\gamma)}$, S_k^P is not dominated by any new expanded paths according to Theorem 4, i.e. S_k^P can not be pruned by PRUNE. In addition, Lemma 1 indicates that $f_l(S_k^P) \leq c^*$. Because of $C \geq c^*$, $f_l(S_k^P) \leq C$, which implies that S_k^P can not be filtered by FILTER. In other words, S_k^P is unchanged, i.e., $S_{k+1}^P = S_k^P$.

2. S_k^P is selected for expansion. When $n_i = \gamma$, $P_{(n_0,\pi(Q),\gamma)}$ is constructed, which is in SG and $G_{cl}(n_i)$ but not in $G_{op}(n_i)$ and $OPEN$. When $n_i \neq \gamma$, the newly constructed sub-path S_{k+1}^P can not be either pruned or filtered according to Theorem 4 and Lemma 1. In other words, S_{k+1}^P is stored into SG, $G_{op}(n_i)$ and in $OPEN$.

Therefore, there is always a sub-path $S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),\gamma)}}$ ($n_i \in Q \cup \{n_0\}$) stored into SG, $G_{op}(n_i)$ and $OPEN$ in each iteration before completing the construction of $P_{(n_0,\pi(Q),\gamma)}$. □

Corollary 1. *A non-shortest path from the start node n_0 to the goal nodes Γ can never be selected for expansion.*

Proof. By contradiction, let $P_{(n_0,\pi(Q),\gamma)}$ ($Q \subseteq N - \Gamma - \{n_0\}$ and $\gamma \in \Gamma$) be a non-shortest path leading to Γ in the graph G with learning effects with cost c'. It is obvious that $c^* < c'$. We have $f_l(P_{(n_0,\pi(Q),\gamma)}) = g_l(P_{(n_0,\pi(Q),\gamma)}) + h_l(\gamma) = g_l(P_{(n_0,\pi(Q),\gamma)}) + 0 = g_l(P_{(n_0,\pi(Q),\gamma)}) = c'$. By Lemmas 1 and 2, there is always a sub-path $S_{(n_0,n_i)}^{P^*}$ of the optimal solution path P^* (n_i is a node on P^*) in $OPEN$ with cost $f_l(S_{(n_0,n_i)}^{P^*}) \leq c^*$ before completing the construction of P^*. There are two cases: (i) Before completing construction of P^*. For the purpose of contradiction, we assume that path $P_{(n_0,\pi(Q),\gamma)}$ is selected before $S_{(n_0,n_i)}^{P^*}$ for expansion. $f_l(P_{(n_0,\pi(Q),\gamma)})$ is the cheapest one in $OPEN$. Therefore, $c' = f_l(P_{(n_0,\pi(Q),\gamma)}) \leq f_l(S_{(n_0,n_i)}^{P^*}) = c^*$, which contradicts $c^* < c'$. (ii) After P^* with $f_l(^*) = c^*$ being constructed. Once $P_{(n_0,\pi(Q),\gamma)}$ is detected, it be filtered because $f_l(P_{(n_0,\pi(Q),\gamma)}) < C = c^*$. □

Theorem 6. *AA^* was admissible.*

Proof. We assume that there is at least one shortest path from n_0 to Γ in a finite labeled directed graph G with learning effects. It is known that all best first search algorithms that prune cycles terminate on finite graphs [33]. Since AA* is a best first search algorithm for finite graphs, a non-shortest solution path is never selected for expansion in terms of Corollary 1. Therefore, an optimal path is returned when AA^* terminates, which means that AA^* is admissible. □

5. AA^* Properties and Efficiency

5.1. Properties of AA^*

Similar to the definition on $c-bounded$ for the A^* algorithm by Pearl [33], we defined $\lambda-bounded$ for the AA^*:

Definition 3. *A path $P_{(n_0,\pi(Q),n_k)}$ ($Q \subseteq N - \Gamma - \{n_0, n_k\}$ and $n_k \in N$) is $\lambda-bounded$ if every sub-path $S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),n_k)}}$ ($n_i \in Q$) satisfies $f_l(S_{(n_0,n_i)}^{P_{(n_0,\pi(Q),n_k)}}) \leq \lambda$.*

From Lemmas 1 and 2 of AA*, which are similar to the corresponding properties of NAMOA*, we have the following similar properties of AA*.

Lemma 3. *Each path $P_{(n_0,\pi(Q),n_i)}$ ($Q \subseteq N - \Gamma - \{n_0, n_i\}$ and $n_i \in N$) selected from $OPEN$ for expansion satisfies that $f_l(P_{(n_0,\pi(Q),n_i)}) \leq c^*$.*

Theorem 7. *A necessary condition for AA^* to select a path P for expansion is that P is $c^*\_bounded$.*

Theorem 8. *A sufficient condition for AA^* to select a path $P_{(n_0,\pi(Q),n_i)}$ ($Q \subseteq N - \Gamma - \{n_0, n_i\}$ and $n_i \in N$) for expansion is that (1) $P_{(n_0,\pi(Q),n_i)}$ be $c^*\_bounded$. (2) $P_{(n_0,\pi(Q),n_i)}$ be a non-dominated path to n_i.*

Theorem 9. *Let $h_l^1(n)$ and $h_l^2(n)$ be two admissible heuristics for the same SPLE problems; AA_1^* and AA_2^* be two versions of algorithm AA* that differ only in the use of heuristic functions $h_l^1(n)$ and $h_l^2(n)$ respectively. If $h_l^1(n) \leq h_l^2(n)$, then all non-dominated and $c^*\_bounded$ paths selected for expansion by AA_2^* would also be selected by AA_1^*.*

5.2. Efficiency and Heuristics

The consistency and monotonicity conditions have a great influence on the traditional A* and NAMOA* algorithms [20]. For example, when either of the two conditions is satisfied, there is no redirect pointer to extended nodes in A* [34]. NAMOA* is optimal among admissible multiobjective algorithms over problems with consistent heuristic functions when efficiency is measured by the number of path expansion operations [20]. Since the cost on each arc changes with its position, we redefine the consistency and monotonicity conditions for the AA* algorithm in the following.

Let $P^r_{(n_i,\pi(Q),n_j)}$ ($Q \subseteq N - \Gamma - \{n_i, n_j\}$ and $n_i \in N$, $n_j \in N$) be the shortest path from n_i to n_j and n_i located at the r^{th} position on a path from n_0 to Γ; $k_l(n_i, n_j, r)$ be the cost of $P^r_{(n_i,\pi(Q),n_j)}$ with $k_l(n_i, n_j, r) = c(n_i, \pi^Q_{[1]}, r+1) + \sum_{i=2}^{|Q|-1} c(\pi^Q_{[i]}, \pi^Q_{[i+1]}, r+i) + c(\pi^Q_{[|Q|]}, n_j, r+|Q|+1)$.

Definition 4. *In a finite labeled directed graph G with learning effects, a heuristic function $h_l(n)$ is $\ell - consistent$ if $h_l(n_i) \leq k_l(n_i, n_j, r) + h_l(n_j)$ ($r = 0, 1, \cdots, \rho$; $r + |Q| + 1 \leq \rho$) holds for each pair of nodes n_i and n_j in G and for each possible position n_i located on the path from n_0 to Γ.*

Definition 5. *In a finite labeled directed graph G with learning effects, a heuristic function $h_l(n)$ is $\ell - monotone$ if $h_l(n_i) \leq c(n_i, n_j, r) + h_l(n_j)$ ($r = 1, 2, \cdots, \rho$) is true for all possible positions at which each arc $(n_i,\ n_j) \in A$ is located on the path from n_0 to Γ.*

According to the above definitions, $\ell - consistent$ and $\ell - monotone$ depend on the related positions r, which make the derivation on $\ell - consistent$ and $\ell - monotone$ not easy. However, the $\ell - consistent$ and $\ell - monotone$ properties in a finite labeled directed graph with learning effects can be deduced from $consistency$ and $monotonicity$ in a finite labeled directed graph without learning effects.

Theorem 10. *If a heuristic function $h(n)$ is $consistent$ in a finite labeled directed graph G without learning effects, then the heuristic function $h_l(n) = \rho^\alpha h(n)$ $(\alpha \leq 0)$ is $\ell - consistent$ in G with learning effects.*

Proof. Let $P_{(n_i,\pi(Q),n_j)}$ be the shortest path from n_i to n_j when node n_i is located at the r^{th} position $(r \leq \rho - (|Q| + 1))$ in a finite labeled directed graph G with learning effects; $k(n_i, n_j)$ be the shortest path cost from node n_i to node n_j in G without learning effects. It can be deduced that:

$$\begin{aligned} k(n_i, n_j) &\leq c(n_i, \pi^Q_{[1]}) + \textstyle\sum_{i=2}^{|Q|-1} c(\pi^Q_{[i]}, \pi^Q_{[i+1]}) + c(\pi^Q_{[|Q|]}, n_j) \\ k_l(n_i, n_j, r) &= c(n_i, \pi^Q_{[1]}, r + 1) + \textstyle\sum_{i=2}^{|Q|-1} c(\pi^Q_{[i]}, \pi^Q_{[i+1]}, r + i) \\ &\quad + c(\pi^Q_{[|Q|]}, n_j, r + |Q| + 1) \end{aligned}$$

$h(n)$ is $consistent$ implies that $h(n_i) \leq k(n_i, n_j) + h(n_j)$. Therefore, $h(n_i) \leq c(n_i, \pi^Q_{[1]}) + \sum_{i=2}^{|Q|-1} c(\pi^Q_{[i]}, \pi^Q_{[i+1]}) + c(\pi^Q_{[|Q|]}, n_j) + h(n_j)$.

Since $\rho > 0$, $\rho^\alpha > 0$, so

$$\rho^\alpha h(n_i) \leq \rho^\alpha c(n_i, \pi^Q_{[1]}) + \rho^\alpha \sum_{i=2}^{|Q|-1} c(\pi^Q_{[i]}, \pi^Q_{[i+1]}) + \rho^\alpha c(\pi^Q_{[|Q|]}, n_j) + \rho^\alpha h(n_j)$$

Similarly, $f(x) = x^\alpha (\alpha < 0)$ is a decreasing function and $r \leq \rho - (|Q| + 1)$, we obtain

$$\begin{aligned} h_l(n_i) &= \rho^\alpha h(n_i) \\ &\leq (r + 1)^\alpha c(n_i, \pi^Q_{[1]}) + \textstyle\sum_{i=2}^{|Q|-1} (r + i + 1)^\alpha c(\pi^Q_{[i]}, \pi^Q_{[i+1]}) \\ &\quad + (r + |Q| + 1)^\alpha c(\pi^Q_{[|Q|]}, n_j) + \rho^\alpha h(n_j) \\ &= k_l(n_i, n_j, r) + h_l(n_j) \end{aligned}$$

Therefore, $h_l(n) = \rho^\alpha h(n)$ $(\alpha < 0)$ is $\ell - consistent$. □

Theorem 11. *If a heuristic function $h(n)$ is $monotone$ in a finite labeled directed graph G without learning effects, then the heuristic function $h_l(n) = \rho^\alpha h(n)$ $(\alpha < 0)$ is $\ell - monotone$ in G with learning effects.*

Proof. $h(n)$ is $monotone$ in a finite labeled directed graph G without learning effects implies that $\forall (n_i, n_j) \in A$, $h(n_i) \leq c(n_i, n_j) + h(n_j)$. Similar to Theorem 10, $h_l(n_i) = \rho^\alpha h(n_i) \leq \rho^\alpha c(n_i, n_j) + \rho^\alpha h(n_j) \leq r^\alpha c(n_i, n_j) + \rho^\alpha h(n_j) = c(n_i, n_j, r) + h_l(n_j)$, which illustrates that $h_l(n) = \rho^\alpha h(n)$ $(\alpha < 0)$ is $\ell - monotone$. □

Theorem 12. *$\ell - monotonicity$ and $\ell - consistency$ are equivalent properties.*

Proof. Let $P_{(n_i,\pi(Q),n_j)}$ be the shortest path from node n_i to node n_j in G with learning effects when node n_i is located at the r^{th} $(r = 0, 1, \cdots, \rho)$ position on a path starting from n_0.

1. $h_l(n)$ is $\ell - monotone$ implies that: (i) $h_l(n_i) \leq c(n_i, \pi^Q_{[1]}, r + 1) + h_l(\pi^Q_{[1]})$; (ii)$h_l(\pi^Q_{[i]}) \leq c(\pi^Q_{[i]}, \pi^Q_{[i+1]}, r + i + 1) + h_l(\pi^Q_{[i+1]})$ $(i = 1, 2, \cdots, |Q| - 1)$; and (iii)$h_l(\pi^Q_{[|Q|]}) \leq c(\pi^Q_{[|Q|]}, n_j,$

$r + |Q| + 1) + h_l(n_j)$. Therefore, $h_l(n_i) \leq c(n_i, \pi^Q_{[1]}, r+1) + \sum_{k=1}^{|Q|-1} c(\pi^Q_{[k]}, \pi^Q_{[k+1]}, r + k + 1) + c(\pi^Q_{[|Q|]}, n_j, r + |Q| + 1) + h_l(n_j) = k(n_i, n_j, r) + h_l(n_j)$, which means that $h_l(n)$ is $\ell - consistent$.

2. $h_l(n)$ is $\ell - consistent$ demonstrates that $h_l(n_i) \leq k_l(n_i, n_j, r) + h_l(n_j)$ for any immediate successor n_j of n_i. Because $k_l(n_i, n_j, r)$ is the cheapest cost from n_i to n_j when node n_i is located at the r^{th} position, $k_l(n_i, n_j, r) \leq c(n_i, n_j, r)$. Therefore, $h_l(n_i) \leq c(n_i, n_j, r) + h_l(n_j)$ for $(n_i, n_j) \in A$, which illustrates that $h_l(n)$ is $\ell - monotone$.

Therefore, $\ell - monotonicity$ and $\ell - consistency$ are equivalent. □

Lemma 4. *If the $\ell - monotonicity$ condition is satisfied, then the f_l values in the search graph are monotonically non-decreasing along every search path.*

Proof. Let $n_{[0]} = n_0 \to n_{[1]} \to n_{[2]} \to \cdots \to n_{[k-1]} \to n_{[k]} = n_j$ ($n_j \in N$) be a path P in the search graph SG. Then arc $(n_{[i-1]}, n_{[i]})$ $(0 < i \leq k)$ is located at the i^{th} position on the path. It follows that $g_l(S^P_{(n_{[0]}, n_{[i]})}) = g_l(S^P_{(n_{[0]}, n_{[i-1]})}) + c(n_{[i-1]}, n_{[i]}, i)$. Therefore, $f_l(S^P_{(n_{[0]}, n_{[i]})}) = g_l(S^P_{(n_{[0]}, n_{[i]})}) + h_l(n_{[i]}) = g_l(S^P_{(n_{[0]}, n_{[i-1]})}) + c(n_{[i-1]}, n_{[i]}, i) + h_l(n_{[i]})$.

Since $\ell - monotonicity$ condition is satisfied, $c(n_{[i-1]}, n_{[i]}, i) + h_l(n_{[i]}) \geq h_l(n_{[i-1]})$. Therefore, $g_l(S^P_{(n_{[0]}, n_{[i-1]})}) + c(n_{[i-1]}, n_{[i]}, i) + h_l(n_{[i]}) \geq g_l(S^P_{(n_{[0]}, n_{[i-1]})}) + h_l(n_{[i-1]}) = f_l(S^P_{(n_{[0]}, n_{[i-1]})})$, i.e., $f_l(S^P_{(n_{[0]}, n_{[i]})}) \geq f_l(S^P_{(n_{[0]}, n_{[i-1]})})$. □

Lemma 5. *Every sub-path $S^{P_{(n_0, \pi(Q), n_i)}}_{(n_0, n_k)}$ ($n_k \in Q$) of the nondominated path $P_{(n_0, \pi(Q), n_i)}$ is a nondominated path.*

Proof. Suppose $S^{P_{(n_0, \pi(Q), n_i)}}_{(n_0, n_k)}$ are dominated by another path to node n_k, $P_{(n_0, \pi(Q_1), n_k)}$ ($Q_1 \subseteq Q$), i.e., $\vec{g}(P_{(n_0, \pi(Q_1), n_k)}) \prec \vec{g}(S^{P_{(n_0, \pi(Q), n_i)}}_{(n_0, n_k)})$. A new path $P(n_0, \pi(Q'), n_i)$ can be generated by combining $P_{(n_0, \pi(Q_1), n_k)}$ and $S^{P_{(n_0, \pi(Q), n_i)}}_{(n_k, n_i)}$. According to Theorem 1, $\vec{g}(P_{(n_0, \pi(Q), n_i)}) \prec \vec{g}(P_{(n_0, \pi(Q'), n_i)})$ which contradicts the non-dominance of $P_{(n_0, \pi(Q), n_i)}$. Therefore, $S^{P_{(n_0, \pi(Q), n_i)}}_{(n_0, n_k)}$ ($n_k \in Q$) is a nondominated path. □

Theorem 13. *If h_l is $\ell - monotone$, necessary conditions for AA^* to select a path $P_{(n_0, \pi(Q), n_i)}$ ($n_i \in N$ and $Q \subseteq N - \{n_0, n_i\} - \Gamma$) for expansion are:*
(1) $P_{(n_0, \pi(Q), n_i)}$ be c^*_bounded.
(2) $P_{(n_0, \pi(Q), n_i)}$ be a non-dominated path to n_i.

Proof. Theorem 7 implies that $P_{(n_0, \pi(Q), n_i)}$ be c^*_*bounded* is a necessary condition.

For the purpose of contradiction, assume that $P_{(n_0, \pi(Q), n_i)}$ is selected for expansion and there exists another nondominated path to n_i, $P_{(n_0, \pi(Q'), n_i)}$, with $\vec{g}(P_{(n_0, \pi(Q'), n_i)}) \prec \vec{g}(P_{(n_0, \pi(Q), n_i)})$. There are two cases:

1. $\vec{g}(P_{(n_0, \pi(Q'), n_i)}) \in G_{op}(n_i) \bigcup G_{cl}(n_i)$, which means that $P_{(n_0, \pi(Q), n_i)}$ would be pruned because $\vec{g}(P_{(n_0, \pi(Q'), n_i)}) \prec \vec{g}(P_{(n_0, \pi(Q), n_i)})$.

2. $\vec{g}(P_{(n_0,\pi(Q'),n_i)}) \notin G_{op}(n_i) \bigcup G_{cl}(n_i)$, i.e., $P_{(n_0,\pi(Q'),n_i)}$ has not been discovered. In terms of Lemma 5, all sub-paths $S_{(n_0,n_k)}^{P_{(n_0,\pi(Q'),n_i)}}$ $(n_k \in Q' \bigcup \{n_0, n_i\})$ are non-dominated. Therefore, they would not be pruned. Since the heuristic is $\ell - monotone$, $f_l(S_{(n_0,n_k)}^{P_{(n_0,\pi(Q'),n_i)}}) \leq f_l(P_{(n_0,\pi(Q',n_i))})$ according to Lemma 4. Similarly, the assumption $\vec{g}(P_{(n_0,\pi(Q'),n_i)}) \prec \vec{g}(P_{(n_0,\pi(Q),n_i)})$ implies that $g_l(P_{(n_0,\pi(Q'),n_i)}) \leq g_l(P_{(n_0,\pi(Q),n_i)})$, i.e., $f_l(P_{(n_0,\pi(Q'),n_i)}) \leq f_l(P_{(n_0,\pi(Q),n_i)})$. $P_{(n_0,\pi(Q),n_i)}$ being selected for expansion demonstrates that $f_l(P_{(n_0,\pi(Q),n_i)}) \leq c^*$ in terms of Lemma 3. Then $f_l(S_{(n_0,n_k)}^{P_{(n_0,\pi(Q'),n_i)}}) \leq f_l(P_{(n_0,\pi(Q),n_i)}) \leq c^* \leq C$. Therefore all sub-paths $S_{(n_0,n_k)}^{P_{(n_0,\pi(Q'),n_i)}}$ can not be filtered, and $P_{(n_0,\pi(Q),n_i)}$ would not be selected for expansion until all sub-paths of $P_{(n_0,\pi(Q'),n_i)}$ are selected for expansion. Therefore, $P_{(n_0,\pi(Q),n_i)}$ would be pruned once $P_{(n_0,\pi(Q'),n_i)}$ is constructed, which contradicts the assumption.

The proof of the theorem is completed. □

Based on Theorem 8 and Theorem 13, we have

Theorem 14. *If h_l is $\ell - monotone$, the necessary and sufficient conditions for AA^* to select a path $P_{(n_0,\pi(Q),n_i)}$ ($n_i \in N$ and $Q \subseteq N - \{n_0, n_i\} - \Gamma$) for expansion are:*
(1) $P_{(n_0,\pi(Q),n_i)}$ be $c^*\_bounded$.
(2) $P_{(n_0,\pi(Q),n_i)}$ be a non-dominated path to n_i.

Let AA_1^* and AA_2^* be two versions of AA* for the same problem that differ only in the use of heuristic functions $h_l^1(n)$ and $h_l^2(n)$ respectively.

Theorem 15. *All paths selected for expansion by AA_2^* will also be selected for expansion by AA_1^* if $h_l^1(n)$ and $h_l^2(n)$ are two admissible heuristics, $h_l^2(n)$ is $\ell - consistent$ and $h_l^1(n) \leq h_l^2(n)$.*

Proof. Theorem 14 illustrates that all paths selected for expansion by AA_2^* are non-dominated and $c^*\_bounded$. For each $c^*\_bounded$ path $P_{(n_0,\pi(Q),n_i)}$ in AA_2^*, $h_l^1(n) \leq h_l^2(n)$ implies that $f_l^1(P_{(n_0,\pi(Q),n_i)}) = g_l(P_{(n_0,\pi(Q),n_i)}) + h_l^1(n) \leq g_l(P_{(n_0,\pi(Q),n_i)}) + h_l^2(n) = f_1^2(P_{(n_0,\pi(Q),n_i)}) \leq c^*$, i.e., all $c^*\_bounded$ paths in AA_2^* are $c^*\_bounded$ in AA_1^*. Therefore, all paths selected for expansion by AA_2^* would be selected for expansion by AA_1^* according to Theorem 8. □

Theorem 16. *Let $h^1(n)$ and $h^2(n)$ be two admissible heuristics for SPPs in a finite labeled directed graph G without learning effects; $h^2(n)$ be $consistent$. $h_l^1(n) = \rho^\alpha h_1(n)$ and $h_l^2(n) = \rho^\alpha h_2(n)$ are heuristics of AA_1^* and AA_2^* respectively. All paths selected for expansion by AA_2^* would also be selected for expansion by AA_1^* if $h^1(n) \leq h_2(n)$.*

Proof. Because $h^1(n)$ and $h^2(n)$ are two admissible heuristics in G without learning effects, $h_l^1(n) = \rho^\alpha h_1(n)$ and $h_l^2(n) = \rho^\alpha h_2(n)$ are admissible in G with learning effects as a result of Theorem 5. Since $h^2(n)$ is $consistent$ in a finite labeled directed graph G without learning effects, $h_l^2(n)$ is $\ell - consistent$ in G with learning effects according to Theorem 10. In addition, since $h_l^1(n) = \rho^\alpha h_1(n)$, $h_l^2(n) = \rho^\alpha h_2(n)$ and $h^1(n) \leq h_2(n)$, then $h_l^1(n) \leq h_l^2(n)$. Based on Theorem 15, all paths selected for expansion by AA_2^* would also be selected for expansion by AA_1^*. □

It is difficult to deduce the $\ell - consistency$ of a heuristic function of AA* in a graph with learning effects. Theorem 16 demonstrates that $\ell - consistency$ could be obtained from the consistency of the heuristic in the graph without learning effects, which makes the derivation much easier. According to Theorems 15 and 16, a greater $h(n)$ or $h_l(n)$ value results in more paths that can be cut down during the search process. Similarly, the estimated cost of the selected path for expansion f_l is not more than c^*. In every iteration, the maximum f_l of all the paths ever being selected for expansion is an approximation to c^*. The more iterations in the AA*, the better the approximation to c^*.

6. Experimental results

Since SPLE has never been studied yet, AA* is compared with the classical backtracking method (which enumerates all paths from the start node to the goal node) in order to illustrate efficiency of the proposed AA*. Similar to [31] [35] [36] , we compare the algorithms on bi-dimensional grids with $s_1 \times s_2$ nodes. Node n_i (identified by its coordinate (x, y)) has several neighbors as successors. Each node on the angle points has two neighbors, each on the edges has three and each inside node has four. Therefore, there are $4 \times s_1 \times s_2 - 2 \times (s_1 + s_2)$ arcs in the grid. Arc costs are randomly generated with a uniform distribution in [1,10]. Assume that the learning index α is -0.2. The involved algorithms search one of the shortest paths from the start node (0,0) to the goal node $(s_1 - 1, s_2 - 1)$ with learning effects. An extreme case is that the longest path has $s_1 \times s_2$ nodes. Since the backtracking method is enumerative which is much time-consuming, we set the maximum 48 hours as the termination criterion, i.e., all the algorithms stop within 48 hours.

In terms of Theorems 15 and 16, $\ell - monotone$ and $\ell - consistency$ heuristic functions exert a great influence on efficiency of AA^*. Therefore, we construct four AA^* algorithms: AA^*_1 with $h^1(n_i) = 0$, AA^*_2 with $h^2(n_i) = \max\{s_1 - 1 - x, s_2 - 1 - y\}$, AA^*_3) with $h_3(n_i) = \sqrt{(s_1 - 1 - x)^2 + (s_2 - 1 - y)^2}$ (Euclidean distance) and AA^*_4 with $h_4(n_i) = (s_1 - 1 - x) + (s_2 - 1 - y)$.

6.1. Comparing AA algorithms against the backtracking method*

To illustrate efficiency of the four constructed AA* algorithms, they are compared to the backtracking method on grid instances with $s_1 \in \{2, 3, 4\}$ and $s_2 \in \{s_2 | s_1 \times s_2 \leq 42 \wedge s_1 \leq s_2 \leq 15\}$. Five instances are generated for each combination of s_1 and s_2. Therefore, 165 instances in total are tested. All algorithms are coded in Visual C++ 2010 and conducted on computers with Windows 7 professional (64 bits), 4G RAM and Intel(R) Core(TM) i5-2400 CPU 3.10 GHz.

Experimental results are analyzed by the multi-factor analysis of variance (ANOVA) technique [37]. A number of hypotheses have to be ideally met by the experimental data. The main three hypotheses (in order of importance) are the independence of the residuals, homoscedasticity or homogeneity of the factor's level variance and normality in the residuals of the model. All the hypotheses are easily accepted since their p-values are zero. The response variable in the experiments is computation time for each algorithm in every instance. Interactions between the compared algorithms and the number of nodes with 95.0% Tukey HSD intervals are shown in Figure 10 and those between the compared algorithms and the number of arcs with 95.0% Tukey HSD intervals are shown in Figure 11.

Figure 10 implies that the AA* algorithms are rather faster than the backtracking method when the number of nodes is more than 16, which is much clearer in the zoomed in area. When the number of nodes is more than 22, the time spent by the backtracking method increases very fast. In fact, the backtracking method cannot finish in 48 hours when the number of node is no less than 44. Similarly, Figure 11 demonstrates that the AA* algorithms are far faster than the backtracking method when the number of arcs is more than 45. The computation time of the backtracking method increases significantly when the number of arcs is more than 90. Therefore, the computation times of the constructed AA* algorithms increase rather slowly with the increase of the number of nodes (arcs) as compared with the backtracking method.

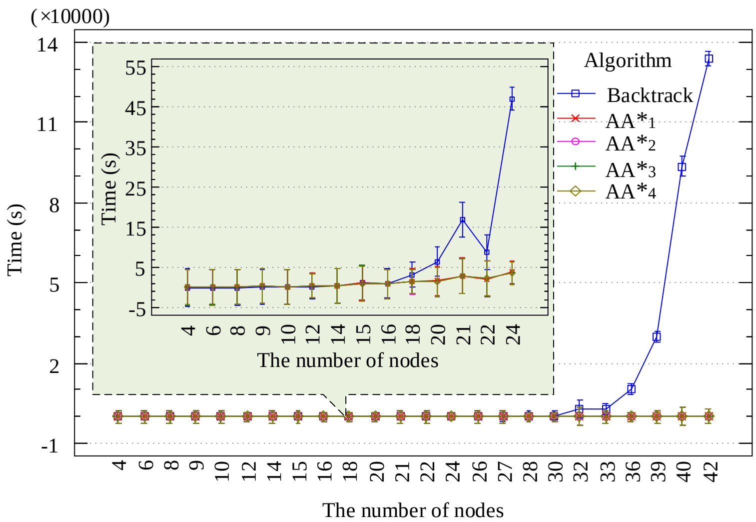

Figure 10: Interactions between algorithms and the number of nodes with 95.0 Percent Tukey HSD intervals

6.2. *Influence of heuristic functions on AA* algorithms*

Though AA* algorithms are more efficient than traditional exact methods (e.g., the backtracking method), different heuristic functions with $\ell - monotone$ and $\ell - consistency$ exert great influences on efficiency of AA* because they result in distinct landscapes. In this subsection, we compare the four constructed AA* algorithms further. Five instances are randomly generated for each grid size $(s \times s)$ where $s \in \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 30\}$, i.e., each of the four AA* algorithms are performed on $14 \times 5 = 70$ grid instances. It is obvious that $h^1(n_i) \leq h_2(n_i) \leq h_3(n_i) \leq h_4(n_i) \leq h^*(n_i)$ where $h^*(n_i)$ is the real cheapest cost from node n_i to the goal node without learning effects in the grids. All these heuristic functions $h_k(n_i)$ $(k = 1, \cdots, 4)$ are consistent heuristics. Since $h_l(n_i) = \rho^\alpha \times h(n_i)$ where $\rho = \min\{|N| - 1, |A|\} = \min\{s^2 - 1, 4s^2 - 4s\}$, $h_l^1(n_i) \leq h_l^2(n_i) \leq h_l^3(n_i) \leq h_l^4(n_i) \leq h_l^*(n_i)$. $h_k(n_i)$ $(k = 1, \cdots, 4)$ are $l - consistent$ heuristics according to Theorem 10. All algorithms are coded in Visual C++ 2010 and carried out on virtual machine with Windows XP professional (32bits) in 1024BM RAM.

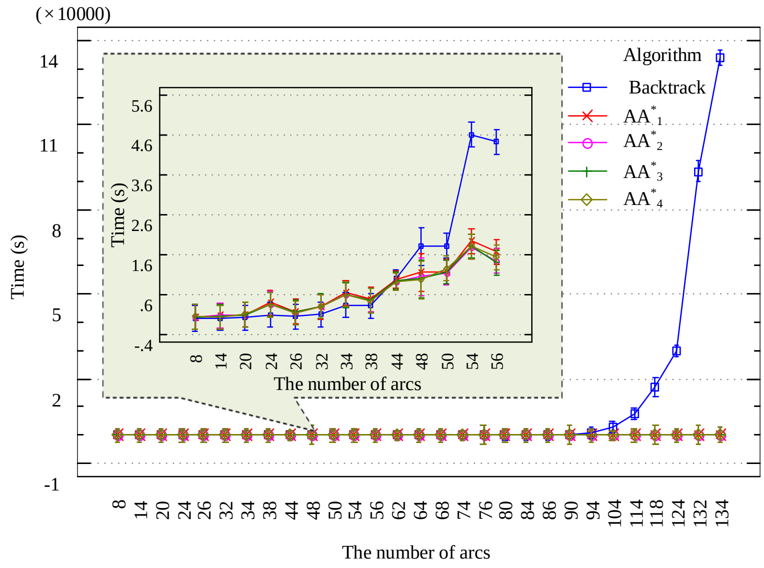

Figure 11: Interactions between algorithms and the number of arcs with 95.0 Percent Tukey HSD intervals

Table 4: Comparisons of the backtracking to the four versions of AA^* on computational time (s)

Grid Size	Node	Arc	Backtracking	AA_1^*	AA_2^*	AA_3^*	AA_4^*
5×5	25	80	36.54	2.00	1.81	1.81	**1.71**
6×6	36	120	6355.35	4.86	4.13	4.03	**3.90**
7×7	49	168	—	11.67	10.51	10.57	**10.22**
8×8	64	224	—	16.85	14.54	14.38	**13.62**
10×10	100	360	—	51.21	44.50	43.33	**42.05**
11×11	121	440	—	91.12	78.97	71.21	**67.21**
12×12	144	528	—	108.60	101.48	99.78	**96.91**
13×13	169	624	—	164.65	155.06	157.64	**148.22**
14×14	196	728	—	235.75	202.88	170.20	**153.74**
15×15	225	840	—	357.58	256.70	247.93	**234.85**
20×20	400	1520	—	1695.81	1388.26	1351.82	**1275.68**
25×25	625	2400	—	5388.04	4653.68	4634.07	**4355.98**
30×30	900	3480	—	44224.16	20917.03	12983.37	**12576.24**

— : the algorithm cannot finish in 48 hours.

Average computation times of the compared algorithms over the five instances for each grid size are shown in Table 4.

From Table 4, it can be observed that the backtracking method for the considered SPLE problems cannot finish in 48 hours when the grid size is no less than $7 \times 7 = 49$, which is in accordance with the results shown in Figure 10. However, all the AA^* algorithms spend only about 10 seconds for the grids with $7 \times 7 = 49$ nodes. When the grid size is less than $8 \times 8 = 64$ nodes, the four AA^* algorithms require similar computation times. For example, the computation times of AA_1^*, AA_2^*,

AA_3^* and AA_4^* are 16.85s, 14.54s,14.38s and 13.62s respectively for $8 \times 8 = 64$ grids. They are not significantly different. However, their computation times are 44224.16s, 20917.03s, 12983.37s and 12576.24s respectively for $30 \times 30 = 900$ grids, of which the differences are rather significant. We can conclude from these that the computation times of AA^* algorithms still increase fast with the increase of the size of grids though the increasing speed is far less than that of enumerative searching methods. On the other hand, AA_1^* requires much more computation time than AA_2^*, AA_3^* and AA_4^* on all instances, e.g., the computation time of AA_1^* is 44224.16s which is more than two times of AA_2^* (20917.03s). AA_2^* needs more time than AA_3^* and AA_4^* on all instances. Though the computation times of AA_3^* and AA_4^* are similar on all instances, AA_4^* always spends less time than AA_3^*, i.e., AA_4^* is the fastest algorithm among the constructed four AA^* algorithms. The reason lies in that the heuristic function $h_l^4(n_i)$ is the closest to $h_l^*(n_i)$, the real cheapest cost from node n_i to the goal node with learning effects in the grid, i.e., a closer heuristic function with $\ell - monotonicity$ and $\ell - consistency$ to the real cheapest cost implies a faster AA^* algorithm.

Furthermore, different learning functions exert influences on performance of AA*. However, the proposed AA* algorithms are still efficient if the h_l function has the same changing rate as that of the learning effect function. For example, the proposal is efficient when $h_l = f(\rho, h)$ if the learning effecting function $c(n_i, n_j, r) = f(r, c(n_i, n_j))$ is concerned.

7. Conclusions and future work

In this paper, we have considered the shortest path problem with learning effects (SPLE), of which each arc has a regularly dynamic cost. The cost of an arc in a path was determined by a function of the arc's position in the path because of learning effects. The shortest sub-paths in SPLE were demonstrated to be unnecessary sub-paths of the final shortest path, which is different from the case in SPP without learning effects. The AA* method was proposed for SPLE problems. A search graph rather than a search tree was adopted to store candidates because there would be more than one candidate sub-path for the final shortest path. With two assumptions, AA* was proven to be admissible. Efficiency of AA* was influenced by the heuristic function with the $\ell - consistency$ and $\ell - monotonicity$ properties which were similar to the $consistent$ or $monotone$ properties in A*. Though it was difficult to judge the $\ell - consistency$ of a heuristic function directly, it could be done by judging consistency of heuristic functions for graphs without learning effects. A closer to (less than) admissible and $\ell - consistent$ heuristic function's real value implies less paths to be expanded. Experimental results illustrated that AA* algorithms were far faster than the backtracking method. Though computation times of AA^* algorithms increased fast with the increase of the size of problems, the increasing speed was far less than those of enumerative searching methods. In addition, a closer heuristic function with $\ell - monotonicity$ and $\ell - consistency$ to the real cheapest cost resulted in faster AA^* algorithms.

In the future, SPLE problems with experience-based or sum-of-processing-time-based learning effects are promising topics. In addition, multi-objective SPLE problems are interesting further work.

References

[1] N. Mathew, S. L. Smith, S. L. Waslander, Planning paths for package delivery in heterogeneous multirobot teams, IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING 12 (2015) 1298–1308.

[2] P. Brass, I. Vigan, N. Xu, Shortest path planning for a tethered robot, Computational Geometry 48 (9) (2015) 732–742.

[3] M. Gendreau, G. Ghiani, E. Guerriero, Time-dependent routing problems: A review, Computers & Operations Research 64 (2015) 189–197.

[4] P. J. Kalczynski, J. Kamburowski, On no-wait and no-idle flow shops with makespan criterion, European Journal of Operational Research 178 (3) (2007) 677–685.

[5] D. Biskup, Single-machine scheduling with learning considerations, European Journal of Operational Research 115 (1) (1999) 173–178.

[6] T. P. Wright, Factors affecting the cost of airplanes, Journal of Aeronautical Sciences 3 (4) (1936) 122–128.

[7] J.-B. Wang, J.-J. Wang, Research on scheduling with job-dependent learning effect and convex resource-dependent processing times, International Journal of Production Research 53 (19) (2015) 5826–5836.

[8] T. Cheng, W.-H. Kuo, D.-L. Yang, Scheduling with a position-weighted learning effect based on sum-of-logarithm-processing-times and job position, Information Sciences 221 (2013) 490–500.

[9] A. Janiak, R. Rudek, A note on a makespan minimization problem with a multi-ability learning effect, Omega 38 (3) (2010) 213–217.

[10] M. B. Hafez, C. K. Loo, Topological q-learning with internally guided exploration for mobile robot navigation, Neural Computing and Applications (2015) 1–16.

[11] X. Ma, Y. Xu, G.-q. Sun, L.-x. Deng, Y.-b. Li, State-chain sequential feedback reinforcement learning for path planning of autonomous mobile robots, Journal of Zhejiang University Science C 14 (3) (2013) 167–178.

[12] A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, A. K. Nagar, A deterministic improved q-learning for path planning of a mobile robot, Systems, Man, and Cybernetics: Systems, IEEE Transactions on 43 (5) (2013) 1141–1153.

[13] W.-C. Yeh, Simplified swarm optimization in disassembly sequencing problems with learning effects, Computers & Operations Research 39 (9) (2012) 2168–2177.

[14] K. KalczynskiPJ, On no-wait and no-idle flowshops with makespan criterion, European Journal of Operational Research 178 (3) (2007) 677–685.

[15] A. Janiak, R. Rudek, Experience-based approach to scheduling problems with the learning effect, Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 39 (2) (2009) 344–357.

[16] D. Biskup, A state-of-the-art review on scheduling with learning effects, European Journal of Operational Research 188 (2) (2008) 315–329.

[17] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, Systems Science and Cybernetics, IEEE Transactions on 4 (2) (1968) 100–107.

[18] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1) (1959) 269–271.

[19] I. Chabini, S. Lan, Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks, Intelligent Transportation Systems, IEEE Transactions on 3 (1) (2002) 60–74.

[20] L. Mandow, J. L. P. De La Cruz, Multiobjective A* search with consistent heuristics, Journal of the ACM (JACM) 57 (5) (2010) 27.

[21] J. T. Thayer, W. Ruml, Bounded suboptimal search: A direct approach using inadmissible estimates, in: The twenty-second International Joint Conference on Artificial Intelligence (IJCAI2011), Vol. 2011, Barcelona, S-pain, 2011, pp. 674–679.

[22] F. W. Trevizan, M. M. Veloso, Depth-based short-sighted stochastic shortest path problems, Artificial Intelligence 216 (2014) 179–205.

[23] S. E. Dreyfus, An appraisal of some shortest-path algorithms, Operations research 17 (3) (1969) 395–412.

[24] D. E. Kaufman, R. L. Smith, Fastest paths in time-dependent networks for intelligent vehicle-highway systems application, Journal of Intelligent Transportation Systems 1 (1) (1993) 1–11.

[25] S. Koenig, M. Likhachev, D. Furcy, Lifelong planning A*, Artificial Intelligence 155 (1) (2004) 93–146.

[26] M. Likhachev, S. Koenig, A generalized framework for lifelong planning A* search., in: Proceedings of the

Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), Monterey, California, USA, 2005, pp. 99–108.

[27] X. Sun, S. Koenig, The fringe-saving A* search algorithm-a feasibility study., in: Twentieth International Joint Conference on Artificial Intelligence (IJCAI2007), Vol. 7, Hyderabad, India, 2007, pp. 2391–2397.

[28] A. Stentz, Optimal and efficient path planning for partially-known environments, in: Robotics and Automation, IEEE International Conference on, IEEE, San Diego, Californi,USA, 1994, pp. 3310–3317.

[29] S. Koenig, M. Likhachev, Fast replanning for navigation in unknown terrain, Robotics, IEEE Transactions on 21 (3) (2005) 354–363.

[30] A. Stentz, The focussed D* algorithm for real-time replanning, in: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI1995), Vol. 95, Qubec, Canada, 1995, pp. 1652–1659.

[31] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, S. Thrun, Anytime search in dynamic graphs, Artificial Intelligence 172 (14) (2008) 1613–1643.

[32] I. Pohl, Heuristic search viewed as path finding in a graph, Artificial Intelligence 1 (3) (1970) 193–204.

[33] J. Pearl, Heuristics, Addison-Wesley Publishing Company Reading, Massachusetts, 1984.

[34] N. J. Nilsson, Artificial Intelligence: A New Synthesis: A New Synthesis, Elsevier, 1998.

[35] L. Mandow, J. P. de la Cruz, A new approach to multiobjective a* search., in: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI2005), Citeseer, Edinburgh, UK, 2005, pp. 218–223.

[36] L. Mandow, J.-L. Pérez de la Cruz, Frontier search for bicriterion shortest path problems, in: Proceedings of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence, IOS Press, Ptras, Greece, 2008, pp. 480–484.

[37] T. Bartz-Beielstein, M. Chiarandini, L. Paquete, M. Preuss, Experimental methods for the analysis of optimization algorithms, Springer, 2010.