Parallel Simplex Method Improvements

Bashir M. Salih *College of Education*
Muhammad W.M. Ali *Directorate of Relations and Information*
University of Mosul, Iraq

Received on: 13/02/2005 **Accepted on: 09/05/2005**

ABSTRACT

The aim of the project is to develop parallel approaches for both the simplex that are used in linear programming to solve linear module systems.

Most of these models are time consuming when executed and processed in the sequential microprocessor computers.

During the project, we try to decrease this time and increase the efficiency of the algorithm for the two methods simplex, through developing parallel methods appropriate to be executed on MIMD type computers.

In this paper, three algorithms were suggested for paralleling (a developed algorithm and a comparison was made between the three algorithms and the original. This comparison is included in second section.

In general, the practical results and the suggested programs for these new algorithms proved to be better in performance than their analogues that are executed in computers of sequential processor in view of the two elements of execution time and algorithm time.

Keywords: simplex method, parallel approaches.

تحسينات الطريقة الفردية المتوازية

بشير محمد صالح *كلية التربية*
محمد واجد محمد علي *مديرية العلاقات والإعلام*
جامعة الموصل

تاريخ استلام البحث: 2005/2/13 **تاريخ قبول البحث: 2005/5/9**

الملخص

هدف البحث هو تطوير طرائق متوازية للطريقة الفردية Simplex Method التي تستعمل في البرمجة الخطية لحل منظومات من النماذج الخطية.

إن معظم هذه النماذج تتطلب وقتاً كبيراً للتنفيذ عند المعالجة باستخدام حاسبات ذات معالج تتابعي، ونحاول في هذا البحث تقليل هذا الوقت وزيادة كفاءة الخوارزميات للطريقة الفردية Simplex Method من خلال تطوير طرائق متوازية ملائمة للتنفيذ على حاسبات من نوع MIMD.

في هذا البحث تم اقتراح ثلاث خوارزميات جديدة للتوازي مطورة نسبة إلى الطريقة الفردية (Simplex Method (SM)) كما تمت المقارنة بين هذه الخوارزميات الثلاث المقترحة مع الخوارزمية الأصلية.

إذ تمكنا من تسريع جميع الطرائق الثلاث المطورة باستخدام التوازي وكان عامل التسريع لإحدى الطرائق المطورة كالآتي:

$$\text{عامل التسريع Sp} = \frac{\text{وقت التنفيذ لـ M1 باستخدام معالج واحد (Ts)}}{\text{وقت التنفيذ لـ M باستخدام P من المعالجات (Tp)}}$$

أي أن:

عامل التسريع = 2.65/ 0.46= 1.92

وعلى العموم أظهرت النتائج العملية والبرمجيات الحاسوبية المقترحة لهذه الخوارزميات الجديدة بأنها افضل من مثيلاتها التي تنفذ على حسابات ذات معالج تتابعي نسبة إلى عنصري زمن التنفيذ وسرعة الخوارزمية.

الكلمات المفتاحية: الطريقة الفردية، الطرائق المتوازية.

1. الطريقة الفردية Simplex Method[4]:

الطريقة الفردية وسيلة رياضية ذات كفاية عالية في إيجاد الحلول المثلى (Optimum Solution) لمشكلات البرمجة الخطية Linear Programming.

لقد تطورت الطريقة الفردية في سنة 1947 وقد قام بتقديم تلك الطريقة وتطويرها الأمريكي جورج دانتز.

ترمي مسألة البرمجة الخطية إلى تعظيم Maximize أو تقليل Minimize دالة هدف خطية معينة وان لدالة الهدف الخطية قيوداً Constraint تحدد مشكلة البرمجة الخطية ولهذه القيود صيغة رياضية معينة فقد يكون القيد أكبر أو يساوي (≥) أو قد يكون أصغر أو يساوي (≤) أو قد يكون القيد مساوياً إلى (=).

وهناك نموذجان رئيسان يمكن كتابة البرمجة الخطية بواسطتهما هما [4] [8]:

1- الصيغة العامة **General Form**

الصيغة العامة لمسألة في البرمجة الخطية هي[8]:

$$\begin{aligned}
&\text{Maximize} && c_1x_1 + c_2x_2 + \ldots + c_nx_n = Z \\
&\text{Subject to} && x_1, x_2, \ldots, x_n \geq 0 \ ; \text{i.e.} \ \ x \geq 0 \\
&\text{and} && a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \ \geq \text{or} = \text{or} \leq b_1 \\
& && a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \geq \text{or} = \text{or} \leq b_2 \quad \ldots\ldots\ldots \quad (1) \\
& && \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
& && a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \ \geq \text{or} = \text{or} \leq b_m
\end{aligned}$$

إذ أن (n) عـــدد المتغيـــرات و(m) عـــدد القيـــود، وأن $(c_1,\ldots,c_n)$ و $(b_1,\ldots,b_m)$ والمصـــفوفة a[i,j] (i=1,..,m وj=1,..,n) تتحـــدد قيمهـــا وطبيعتهـــا وفقـــاً للمشـــكلة المرغـــوب حلها.

2-الصيغة القياسية **Standard Form**

إن الصيغة القياسية هي إضافة متغيرات غير سالبة على الصيغة العامة (1) أو حذفها اعتماداً على علامة المتباينة بحيث تصبح كما يأتي[8]:

$$\begin{aligned}
&\text{Maximize} && {}_1cx_1 + c_2x_2 + \ldots + c_nx_n = Z \\
&\text{Subject to} && x_1, x_2, \ldots, x_n , x_{m+1}, \ldots, x_{m+n} \geq 0 \\
&\text{And} && a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + x_{m+1} = b_1 \\
& && a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + x_{m+2} = b_2 \\
& && \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
& && a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n + x_{m+n} = b_m
\end{aligned}$$

إن أي حل أساسي أولي هو الحل الذي يبدأ من نقطة الأصل، أي ان الحل الأولي هو الحل الناتج حينما تكون القيم:

$$x_1 = x_2 = \ldots = x_n = 0$$

ولهذا فان: $x_{m+1} = b_1 , x_{m+2} = b_2 , \ldots , x_{m+n} = b_m$

الهـدف هـو تحويـل هـذه المتغيـرات x_n ,... ,x_2 ,x_1 الـى المتغيـرات الأساسـية Basic Variables، ويطلق على المتغيرات x_{m+n} ,...,x_{m+2} ,x_{m+1} المتغيرات غير الأساسية Non-Basic Variable. ويمكـن تمثيـل البيانـات السـابقة بشـكل جـدول لكـي يسـهل علينـا فهـم المتغيرات الأساسية وغير الأساسية:

جدول (1) طريقة تمثيل المتغيرات الأساسية وقيم b_i ومعاملات $x_1,..,x_n$ وقيمة دالة الهدف ومعاملاتها للطريقة الفردية [8].

Basic	Value	x_1	x_2	...	x_{m+1}	...	x_n
x_1	b'_1	a_{11}	a_{12}	:	a_{1m}	:	a_{1n}
x_2	b'_2	a_{21}	a_{22}	:	a_{2m}	:	a_{2n}
:	:	:	:	:	:	:	:
x_m	b'_m	a_{m1}	a_{m2}	:	a_{mm}	:	a_{mn}
-z	$-z'_o$	c_1	c_2	.	c_m	.	c_n

2.خطوات حل الطريقة الفردية [10]:

1- الخطوة الأولى: إيجاد المتغير الذي سيدخل الحل بوصفه متغيراً أساسياً:

نسمي المتغيرات $x_{m+1}, ..., x_n$ بالمتغيرات غير الأساسية، نحاول إيجاد أقل عنصر في $c_1, ..., c_s,..., c_n$ وليكن c_s ، فإذا كان $c_s<0$؛ إذن بزيادة x_s سوف تزداد قيمة دالة الهدف Z، أما إذا كان هناك اكثر من عنصر $c_j<0$ فاننا نختار العنصر الأكثر سالبيةc_s؛ لأنه في هذه الحالة سوف تزداد قيمة دالة الهدف أسرع، وبهذا يكون العنصر x_s هو العنصر الداخل متغيراً أساسياً.

2- الخطوة الثانية: إيجاد العنصر الذي سيخرج من الحل بوصفه متغيراً غير أساسي:

في القيد (i)، إذا كان $a_{is}>0$ القيمة الأكبر؛ فتكون قيمة x_s هي b_i/a_{is}، وإلا سيكون x_i سالباً. إذن سيكون x_s هو الذي سيزيد القيمة

$$\max x_s = \min_{\substack{i=1,...,m \\ a_{is}>0}} \left(\frac{b_i}{a_{is}}\right) \qquad (2)$$

فإذا وجدت هذه القيمة في الصف r، فإن $x_r = 0$ عندما x_s يعطى بالقيمة b_r/a_{rs}، أما بقية المتغيرات الأساسية فسوف تبقى موجبة. العنصر a_{rs} يسمى بالعنصر المحوري، والعمود s هو العمود المحوري، والصف r هو الصف المحوري.

3- الخطوة الثالثة: الحصول على الصيغة الرسمية الجديدة:

المتغيرات الأساسية الجديدة هي $x_1,x_2,...,x_s,..x_m$، المتغير x_s سوف يكون متغيراً غير أساسي. للحصول على صيغة جديدة نجري العمليات المحورية للتخلص من المتغير x_s في جميع القيود إلا دالة الهدف والقيد المحوري.

تعاد هذه الخطوات إلى أن تكون العناصر في دالة الهدف كلها موجبة، بهذا نكون قد وصلنا إلى الحل الأمثل ولا يوجد حل أفضل منه.

ويمكن تمثيل هذه الخطوات عن طريق الجدول الخاص بالطريقة الفردية [8، فاذا كان لدينا النظام (1) فيمكن تحويله إلى الجدول الآتي:

الجدول (2) طريقة تمثيل المتغيرات الأساسية وقيم b_i ومعاملات $x_1,..,x_n$ ودالة الهدف ومعاملاتها للطريقة الفردية، بالنسبة للتكرار k. [8]

Ite.	Basic	Value	x_1	x_2	...	x_r	...	x_m	x_{m+1}	...	x_s	...	x_n
k	x_1	b'_1	1	:	:	:	:	:	a'_{1m+1}		a'_1		a'_{1n}
	x_2	b'_2		1	:	:	:	:	a'_{2m+1}		a'_{2s}		a'_{2n}
	:	:	:	:	:	:	:	:	:	:	:	:	:
	x_r	b'_r	:	:	:	1	:	:	a'_{rm+1}	:	a'_{rs*}	:	a'_{rn}
	:	:	:	:	:	:	:	:	:	:	:	:	
	x_m	b'_m	:	:	:	:	:	1	a'_{mm+1}	:	a'_{ms}	:	a'_{mn}
	-z	$-z'_o$	.	.	.	.	.	.	c'_{m-1}	:	c'_m	:	c'_n

هذا في التكرار k وعند إجراء الخطوات الثلاث المذكورة آنفاً نحصل على التكرار (k+1) وهو:

الجدول (3) طريقة تمثيل المتغيرات الأساسية وقيم b_i ومعاملات $x_1,..,x_n$ ودالة الهدف ومعاملاتها للطريقة الفردية، بالنسبة إلى التكرار k+1. [8]

Ite.	Basic	Value	x_1	x_2	...	x_r	...	x_m	x_{m+1}	...	x_s	...	x_n
K+1	x_1	b^+_1	1	:	:	:	:	:	$+_{1m+1}a$	:	:	:	$+_{1n}a$
	x_2	$+_2 b$		1	:	:	:	:	$+_{2m+1}a$	:	:	:	$+_{2n}a$
	:	:	:	:	:	:	:	:	:	:	:	:	:
	x_r	$+_r b$	:	:	:	1	:	:	$+_{rm+1}a$	:	1	:	$+_{rn}a$
	:	:	:	:	:	:	:	:	:	:	:	:	
	x_m	$+_m b$	:	:	:	:	:	1	$+_{mm+1}a$	:	:	:	$+_{mn}a$
	-z	$+_o -z$	.	.	.	.	.	.	$+_{m-1}c$	:	.	:	$+_n c$

إذ أن

$$
\left.
\begin{array}{ll}
b^{+}_{r} = b'_{r} / a'_{rs} & \text{.....(a)} \\
a^{+}_{rj} = a'_{rj} / a'_{rs} & \text{.....(b)} \\
b^{+}_{i} = b'_{i} - a'_{rs} b^{+}_{r} & \text{.....(c)} \\
a^{+}_{ij} = a'_{ij} - a'_{is} a^{+}_{rj} \quad (i \neq r) & \text{.....(d)} \\
c^{+}_{j} = c'_{j} - c'_{s} a^{+}_{rj} & \text{.....(e)} \\
z^{+}_{o} = z'_{o} + c'_{s} b^{+}_{r} & \text{.....(f)}
\end{array}
\right\} \quad (3)
$$

مثال تطبيقي [8]:

جد أقل قيمة لـ:

$$-2x_1 - 4x_2 = Z$$

بحيث أن

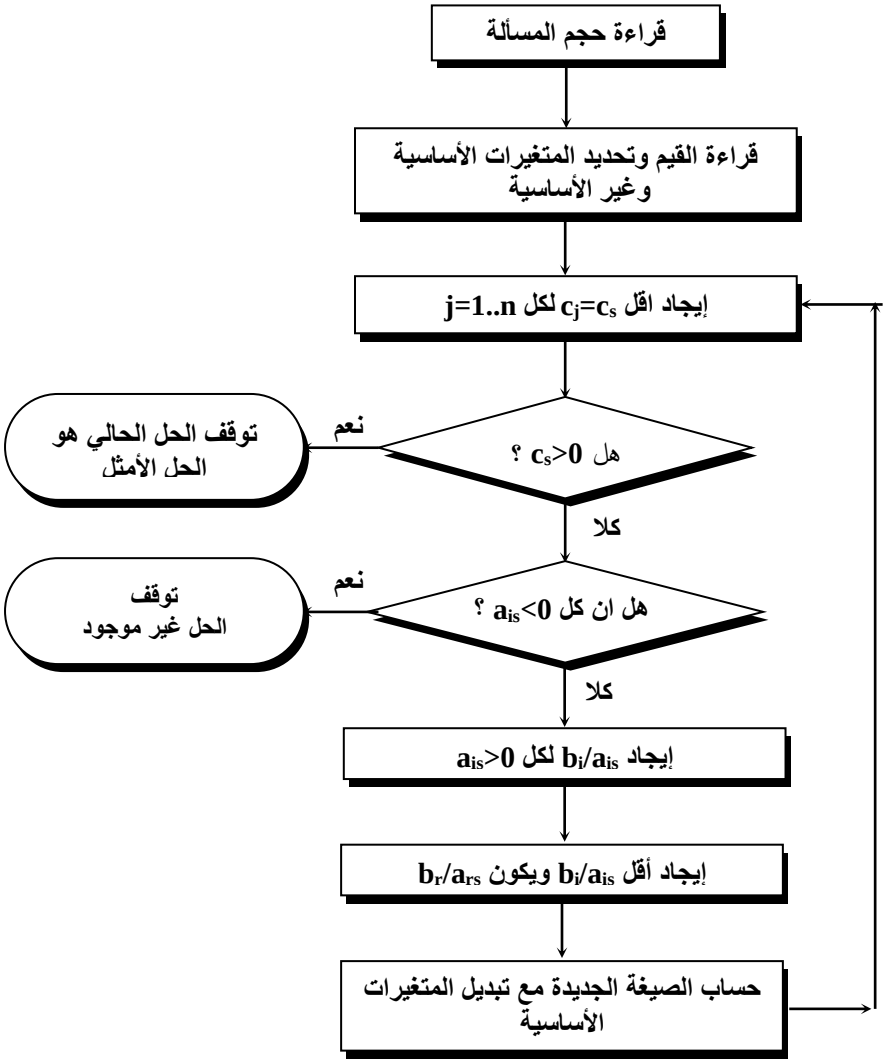
$$x_1, x_2 \geq 0$$

$$3x_1 + 4x_2 \leq 1700$$

$$2x_1 + 5x_2 \leq 1600$$

3. برمجة الطريقة الفردية:

من الممكن تنفيذ الطريقة الفردية على الحاسبة وذلك باتباع الخطوات الموضحة في المخطط الانسيابي الآتي [8]:



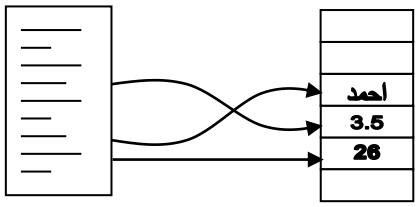
الشكل (1): المخطط الانسيابي للطريقة الفردية الاعتيادية

4. التوازي في الطريقة الفردية:

سنناقش في هذا البند ثلاث طرائق متوازية جديدة تم التوصل إليها من خلال دراستنا لموضوع التوازي للطريقة الفردية، وبما إننا نتكلم على الحاسبات المتوازية فلا بد من التطرق إليها:

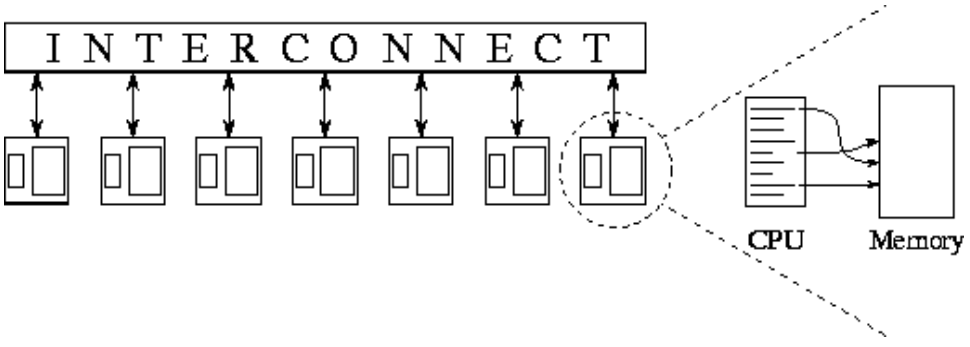
4-1 الحاسبات المتوازية:

يعود الفضل للتقدم السريع في صناعة الحاسبات ودخولها مجالات عدة مثل التجارة والعلوم والتعليم لنموذج الحاسبة الأولى (Single Machine Model) لحاسبات Von Neuman، وكما هو معروف فان هذه الحاسبات تمتلك وحدة معالجة مركزية واحدة (CPU) مرتبطة بوحدة خزن (Memory) (شكل1)، وان المعالج يقوم بتنفيذ برنامج مخزون والذي يحدد بواسطة مجموعة متسلسلة من عمليات القراءة والكتابة على الذاكرة [15].



الشكل (1) حاسبة Von Neuman، تقوم وحدة المعالجة المركزية (CPU) بانجاز البرنامج الذي ينفذ بشكل تسلسلي لعمليات القراءة والكتابة على الذاكرة المحجوزة [15].

بينما في الحاسبات المتوازية فهو ايجاد نموذج متوازٍ يتكون من عدد من حاسبات Von Neuman (الشكل 2) يكون عاماً لكثير من التطبيقات وان تمتلك الحاسبة المتوازية ميزتين اساسيتين هما (البساطة والواقعية)، والبساطة تعني امكانية فهم الحاسبة والبرمجة لها، والواقعية هي ضمان تنفيذ النماذج البرمجية لها بكفاية معقولة [6].



الشكل (2): نموذج حاسبة متوازية مثالية، تحتوي كل نقطة ارتباط على حاسبة من نوع Von Neuman ، ويمكن لكل نقطة الاتصال بالنقاط الأخرى عن طريق إرسال الرسائل واستقبالها عبر الشبكة المربوطة. [15]

2-4 التوازي

الحاسوب المتوازي هو مجموعة من المعالجات التي بامكانها العمل بصورة متوازية او تعاونية (Cooperatively) لحل مسألة حسابية ما، وهذا التعريف يشمل الحاسبات المتوازية التي تمتلك عشرات او مئات من المعالجات (Processors) [6].

3-4 الخوارزمية المتوازية:

الخوارزمية: هي مجموعة من التوجيهات لتنفيذ عمليات حسابية مصممة بشكل يؤدي الى حل المسألة المعطاة [1]، [2].

وعليه فمن الممكن تعريف الخوارزمية المتوازية: بأنها تجزئة مسألة واحدة إلى عدة مسائل جزئية مستقلة حتى يمكن حل كل مسألة جزئية في معالج مستقل ثم جمع الحلول لتركيب حل المسألة الأصلية [7].

4-4 تصنيف الحاسبات المتوازية:

يعمل أي نوع من أنواع الحاسبات (حاسبات تسلسلية كانت أم متوازية) على تنفيذ مجموعة من الإيعازات (Instructions) على مجموعة من البيانات (Data) ومن ثم تتم معالجتها وإخراجها على شكل معلومات[6].

وهناك تصانيف عدة لمعمارية الحاسبات، من اشهر هذه التصانيف تصنيف الباحث Flynn [9]، فقد صنف أنظمة بحسب نوع التحكم (Type of Control) والخوارزميات (Algorithms) والمعالجات المنطقية (Logical Processors) إلى أربعة أصناف رئيسة [3]، [6]، [13]، [14] هي:

1-معالجات ذات إيعاز فردي وبيانات جارية مفردة

SISD: Single Instruction Stream, Single Data Stream.

2- معالجات ذات إيعازات متعددة وبيانات جارية مفردة.

MISD: Multiple Instruction Stream, Single Data Stream.

3- معالجات ذات إيعاز فردي وبيانات جارية متعددة.

SIMD: Single Instruction Stream, Multiple Data Stream.

4- معالجات ذات إيعازات متعددة وبيانات جارية متعددة.

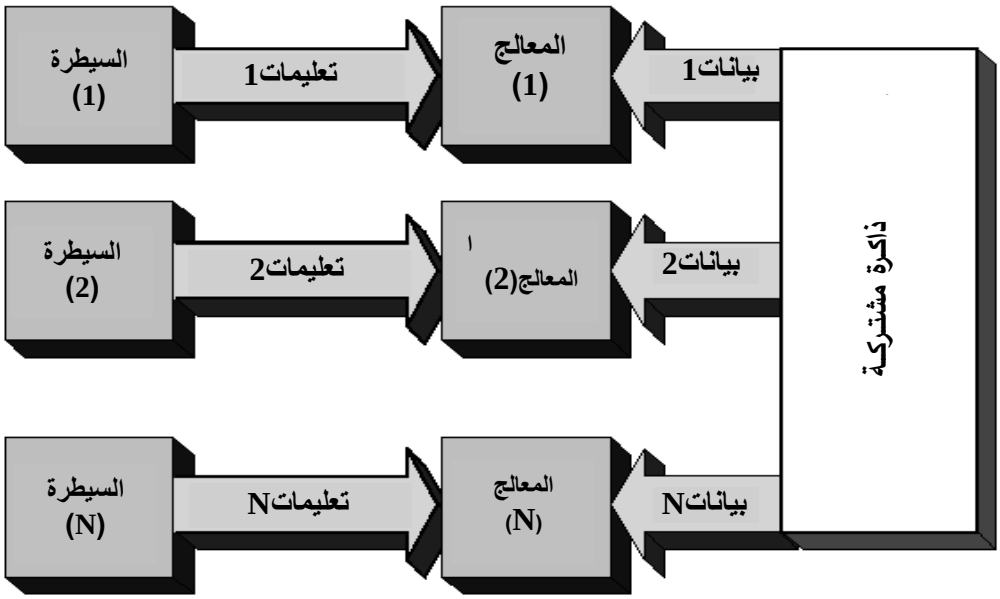
MIMD: Multiple Instruction Stream, Multiple Data Stream.

وهدفنا من البحث هو تطوير طرائق متوازية ملائمة للتنفيذ على حسابات من نوع MIMD ولهذا سوف نقتصر الحديث عنها:

5-4 حاسبات من نوع MIMD [5]، [6]، [11]

يعد هذا النوع من الحاسبات أكثر شيوعاً وكفاية وقوة في تنفيذ التطبيقات العامة في موضوع الحاسبات المتوازية اذ تكون فيها مجموعة الايعازات ومجموعة البيانات مختلفة.

يحتوي هذا النوع من الحاسبات على (N) من المعالجات وعلى (N) من الايعازات وعلى (N) من البيانات الشكل (3). ويمتلك كل معالج ذاكرة خاصة به فضلاً عن وحدة الحساب والمنطق ووحدة السيطرة الخاصة به، وله القدرة على العمل بصورة ليست متزامنة ومستقلة على بياناته الجارية الخاصة، وفي اي وقت من الممكن ان تقوم معالجات مختلفة بتنفيذ ايعازات مختلفة وعلى بيانات مختلفة [3] ,[12].



الشكل(3): مخطط توزيع الإيعازات والبيانات في حاسبات من نوع MIMD [14]

نأتي الآن على شرح الطرائق الثلاث التي تم التوصل اليها:

6-4 الطريقة المتوازية الجديدة الأولى:

كما لاحظنا خطوات حل الطريقة الفردية؛ فإن هناك خطوات مستقلة فيما بينها أي أنه من الممكن أن ننجزها في الوقت نفسه؛ لأنها لا تعتمد على بعضها البعض، فإذا أخذنا المعادلات المستخدمة في حل الطريقة (3) من الممكن كتابتها بالشكل الآتي:

$$
\begin{aligned}
& b^{+}_{r} = b'_{r} / a'_{rs} \quad \text{..... (a)} \\
& z^{+}_{o} = z'_{o} + c'_{s}\, b^{+}_{r} \quad \text{..... (b)} \\
& b^{+}_{i} = b'_{i} - a'_{rs}\, b^{+}_{r} \;;\; i \neq r \quad \text{..... (c)} \qquad (4) \\
& a^{+}_{rj} = a'_{rj} / a'_{rs} \quad \text{..... (d)} \\
& a^{+}_{ij} = a'_{ij} - a'_{is}\, a^{+}_{rj} \;;\; i \neq r \quad \text{..... (e)} \\
& c^{+}_{j} = c'_{j} - c'_{s}\, a^{+}_{rj} \quad \text{..... (f)}
\end{aligned}
$$

من الملاحظ ان المعادلتين (b) و(c) تعتمدان على المعادلة (a) أي انه يجب حساب (b^{+}_{r}) الذي يدخل في حساب المعادلتين (b) و(c)؛ لهذا فإن المعادلات (a–c) ستكون على معالج واحد، أما المعادلتان (e) و(f)؛ فانهما تعتمدان على قيمة (a^{+}_{rj}) المحسوب في المعادلة (d) فيجب وضع

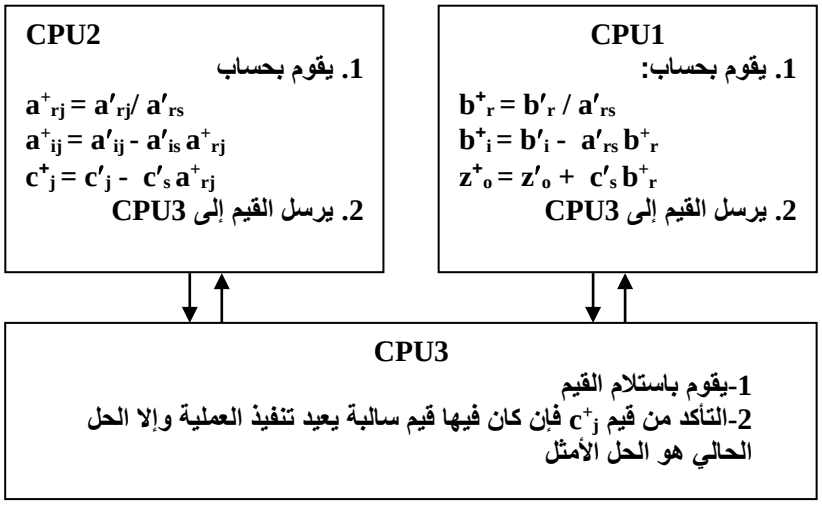
المعادلات (f–d) في معالج آخر، لأن كلاً من المجموعتين (المعادلات (c–a)) و(المعادلات (d–f)) لا تعتمد الواحدة على الأخرى، وبذلك نكسب التوازي في هذه النقطة.

فإذا مثلنا الطريقة الجديدة على الجدول المستخدم في حل المسألة لتبين لنا التوازي بشكل أوضح كما في الجدول (4):

الجدول (4): العمليات المستقلة التي تجرى على الجدول الخاص بالطريقة الفردية.

Basic	**Value**	x_1	x_2	…	x_r	…	x_m	x_{m+1}	…	x_s	…	x_n
x_1	b^+_1	1	:	:	:	:	:	a^+_{1m+1}	:	:	:	a^+_{1n}
	CPU1							**CPU2**				**CPU2**
x_2	b^+_2		1	:	:	:	:	a^+_{2m+1}	:	:	:	a^+_{2n}
	CPU1							**CPU2**				**CPU2**
:	:	:	:	:	:	:	:	:	:	:	:	:
x_r	b^+_r	:	:	:	1	:	:	a^+_{rm+1}	:	1	:	a^+_{rn}
	CPU1							**CPU2**				**CPU2**
:	:	:	:	:	:	:	:	:	:	:	:	
x_m	b^+_m	:	:	:	:	:	1	a^+_{mm+1}	:	:	:	a^+_{mn}
	CPU1							**CPU2**				**CPU2**
-z	$-z^+_o$	.	.	.	.	.	.	c^+_{m-1}	:	.	:	C^+_n
	CPU1							**CPU2**				**CPU2**

ويمكن تمثيل الطريقة المتوازية بالشكل الآتي:



الشكل (4) : توزيع الأوامر على المعالجات في الطريقة الأولى للتوازي في الطريقة الفردية

1-6-4 توزيع الأعمال على المعالجات للطريقة المتوازية الأولى:

يمكن توضيح عمل المعالجات بالشكل الآتي:

1-الـ CPU1: يقوم بحساب الآتي:

قيم b^{+}_{I} الجديدة وقيم دالة الهدف الجديدة أي العمود Value في الجدول الخاص بالطريقة الفردية، ويرسل القيم إلى CPU3 .

2-الـ CPU2: يقوم بحساب الآتي:

قيم a^{+}_{rj} وa^{+}_{ij} وc^{+}_{j} الجديدة ويرسل القيم إلى CPU3 .

3-الـ CPU3: يقوم بحساب الآتي:

يقوم باستلام القيم، التأكد من قيم c^{+}_{j} فإن كانت فيها قيم سالبة يعيد تنفيذ العملية وإلا فإن الحل الحالي هو الحل الأمثل.

4-6-2 حساب عامل التسريع للطريقة الأولى

وكما ذكرنا سابقاً إن حاسبات من نوع MIMD غير متوافرة في بلادنا، فلذلك تم تنفيذ كل إجراء على حده وحساب الوقت كذلك ويكون الوقت الأكبر هو الوقت المعتمد في حساب تنفيذ الإجراءين. وقد نفذ البرنامج باستخدام لغة Turbo Pascal 7.0 وحاسبة PIII ذات معالج 733MHz وباستخدام دالة Delay (120000) فكانت المدة المستغرقة هي:1.52 Sec

فعند حساب عامل التسريع Speed up بين الطريقة العادية والطريقة المتوازية الأولى نحصل على (5)(6):

$$\text{التسريع Sp} = \frac{\text{وقت التنفيذ لـM1 باستخدام معالج واحد (Ts)}}{\text{وقت التنفيذ لـM باستخدام N من المعالجات (Tp)}}$$

$$\text{التسريع} = \frac{2.65}{1.52} = 1.74$$

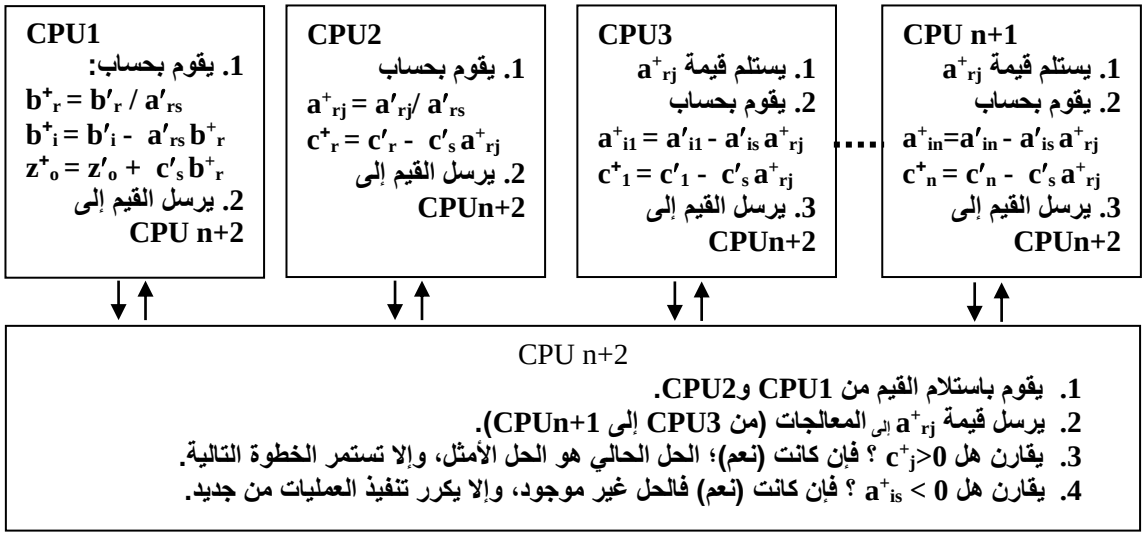
إذن كما نلاحظ ان عامل التسريع الذي حصلنا عليه هو (1.74) أي أن مقدار التسريع في تنفيذ البرنامج قد زاد وبذلك نكون قد كسبنا وقتاً جيداً للتنفيذ على الرغم من أن المثال المستخدم في تنفيذ البرنامج يعتمد على متغيرات قليلة نوعاً ما، أما إذا كانت هناك متغيرات كبيرة فإن عامل التسريع سيكون بشكل أكبر.

4-7 الطريقة المتوازية الجديدة الثانية:

من الممكن تطوير طريقة جديدة للتوازي في الطريقة الفردية، فلو لاحظنا الطريقة السابقة لرأينا أنه بالإمكان زيادة عدد المعالجات المستخدمة، حتى يكون لكل عمود في جدول الحل معالجه الخاص به، وينفذ جميع العمليات الموجودة في العمود.

ففي هذه الحالة سوف نحتاج إلى عدد من المعالجات يساوي (n+2) (اذ ان n يمثل عدد الأعمدة في الجدول الخاص بالطريقة الفردية) على العكس من الطريقة الأولى التي نحتاج فيها فقط إلى ثلاث معالجات مهما كان كبر حجم المسألة التي عندنا، ومن هنا يتضح لنا الفرق بين الطريقة الأولى والطريقة الثانية، إذ أنه يتم استخدام الطريقة الأولى عندما يكون عدد المعالجات محدوداً والمسألة عندنا صغيرة نسبياً، أما في حالة كون عدد المعالجات كبيراً والمسألة فيها قيود كثيرة فإننا نستخدم الطريقة الثانية المطورة.

ومن الممكن تمثيل الطريقة المطورة عن طريق المخطط الآتي:



الشكل(5): يمثل توزيع الأوامر على المعالجات في الطريقة الثانية للتوازي في الطريقة الفردية.

4-7-1 توزيع الأوامر على المعالجات للطريقة المتوازية الثانية:

يمكن توضيح عمل المعالجات بالشكل الآتي:

الـ CPU1: يقوم بحساب الآتي:

قيم b^+_I الجديدة وقيم دالة الهدف الجديدة أي العمود Value في الجدول الخاص بالطريقة الفردية، ويرسل القيم إلى CPUn+2.

الـ CPU2: يقوم بحساب الآتي:

قيم a^+_{rj} وc^+_r الجديدة ويرسل القيم إلى CPUn+2.

الـ CPU3: يقوم بحساب الآتي:

يقوم باستلام قيمةa^+_{rj} ، وبعد ذلك يقوم بحساب قيمة $\mathbf{a^+_{i1}}$ و قيمة c^+_1 أي يحسب العمود الأول.

الـ CPUn+1: يقوم بحساب الآتي:

يقوم باستلام قيمةa^+_{rj}، وبعد ذلك يقوم بحساب قيمة $\mathbf{a^+_{in}}$ و قيمة c^+_n أي يحسب العمودn.

الـ CPUn+2: يقوم بحساب الآتي:

يقوم باستلام قيم b^+_I وقيمةa^+_{rj} ، وبعد ذلك يقوم بإرسالها إلى المعالجات الأخرى وبعد ذلك يقارن هل $c^+_j>0$ ؟ فإن كانت (نعم) الحل الحالي هو الحل الأمثل، وإلا تستمر الخطوة التالية التي يقوم بها بالمقارنة هل $a^+_{is} < 0$ ؟ فإن كانت (نعم) فالحل غير موجود، وإلا يكرر تنفيذ العمليات من جديد.

4-7-2 حساب عامل التسريع للطريقة الثانية

الوقت المستغرق لتنفيذ العمليات المحورية –بالنسبة إلى المثال– هو 0.22 ثانية، مضافاً إليه الوقت الذي يستغرقه المعالج السادس (CPU6) ويقدر بـ(0.24) ثانية فيكون الوقت المستغرق لكل المعالجات هو:

$(0.22 + 0.24)*3 = 1.38sec$

وعليه فإن عامل التسريع سيكون كبيراً، ومن الممكن حسابه عن طريق:

$$\text{التسريع Sp} = \frac{\text{وقت التنفيذ لـM1 باستخدام معالج واحد (Ts)}}{\text{وقت التنفيذ لـM باستخدام P من المعالجات (Tp)}}$$

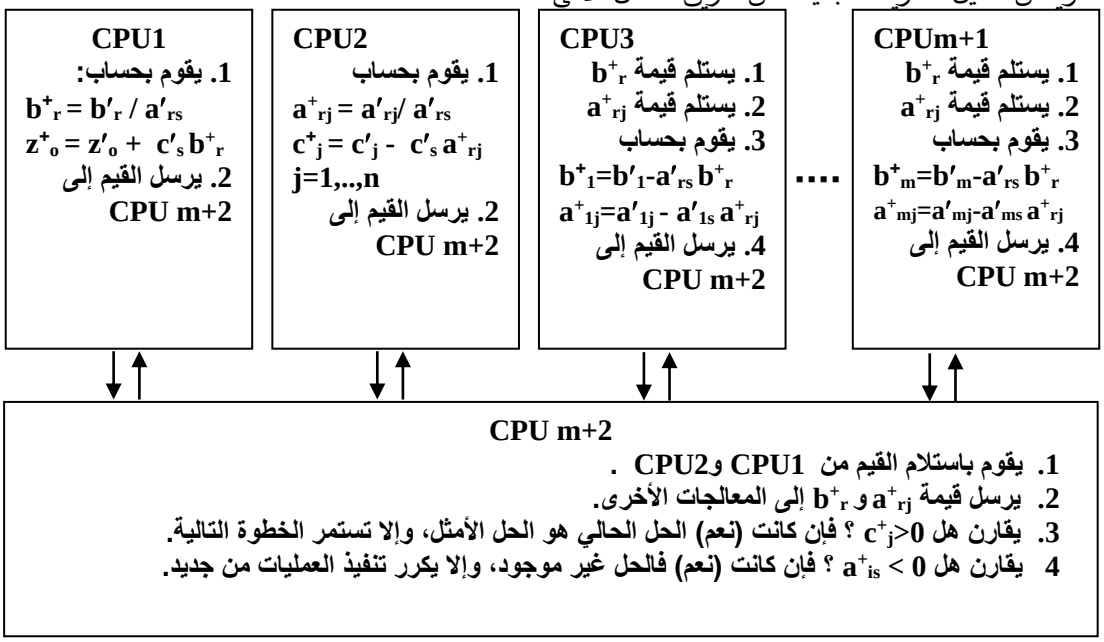
التسريع = 2.65 / 0.46 = 1.92

إذن فان الطريقة الثانية تكون أسرع من الطريقة الأولى؛ لأن عامل التسريع فيها أكبر، أما من ناحية عدد المعالجات فإن عدد المعالجات في الطريقة الثانية هو أكبر من الطريقة الأولى وهذا يعني زيادة في الكلفة، إذن عندما تكون لدينا مسألة فيها قيود كثيرة فمن المستحسن استخدام الطريقة الثانية أما إذا كان لدينا مسألة فيها قيود قليلة؛ فإن الطريقة الأولى تكون هي الأفضل.

8-4 الطريقة المتوازية الجديدة الثالثة:

في حالة وجود مسألة تكون فيها عدد القيود أكبر من عدد المتغيرات، فالطريقة الثانية لا تكون مناسبة للعمل؛ لأن ظاهرة التوازي فيها تكون قليلة، فللتخلص من هذه المشكلة نقترح طريقة أخرى للتوازي، وذلك عن طريق الصفوف من الجدول الخاص بحل الطريقة الفردية. إذ يمكن تقسيم المعالجات على القيود، فمثلاً المعالج الأول ينفذ العمليات التي تتم في القيد الأول، والمعالج الثاني ينفذ العمليات التي تتم في القيد الثاني.. وهكذا. في الطريقة السابقة كنا نحتاج إلى عدد من المعالجات يساوي (n+2). أما في هذه الطريقة فإننا سوف نحتاج إلى (m+2) من المعالجات.

ويمكن تمثيل الطريقة الجديدة عن طريق الشكل الآتي:



الشكل (6): توزيع الأوامر على المعالجات في الطريقة الثالثة للتوازي في الطريقة الفردية.

1-8-4 توزيع الأوامر على المعالجات للطريقة المتوازية الثالثة

ويمكن توضيح عمل المعالجات بالشكل الآتي:

الـ CPU1: يقوم بحساب الآتي:

قيم b^{+}_{r} الجديدة وقيم دالة الهدف الجديدة أي العمود Value في الجدول الخاص بالطريقة الفردية، ويرسل القيم إلى CPUm+2.

الـ CPU2: يقوم بحساب الآتي:

قيم a^{+}_{rj} وc^{+}_{j} (لكل j) الجديدة ويرسل القيم إلى CPUm+2 .

الـ CPU3: يقوم بحساب الآتي:

استلام قيم b^{+}_{r} و a^{+}_{rj} ، وبعد ذلك يقوم بحساب قيم b^{+}_{1} و a^{+}_{1j} أي يحسب الصف الأول.

الـ CPUm+1: يقوم بحساب الآتي:

استلام قيم b^{+}_{r} و a^{+}_{rj} ، وبعد ذلك يقوم بحساب قيم b^{+}_{m} و a^{+}_{mj} أي يحسب الصف m.

الـ CPUm+2: يقوم بحساب الآتي:

يقوم باستلام قيمة b^{+}_{r} من الـCPU1 وقيمةa^{+}_{rj} من الـCPU2، وبعد ذلك يقوم بارسالها إلى المعالجات الأخرى وبعد ذلك يقارن هل $c^{+}_{j}>0$ ؟ فإن كانت (نعم) الحل الحالي هو الحل الأمثل، وإلا تستمر الخطوة التالية، التي يقوم بها بالمقارنة هل $a^{+}_{is}<0$ ؟ فإن كانت (نعم) فالحل غير موجود، وإلا يكرر تنفيذ العمليات من جديد.

2-8-4 حساب عامل التسريع للطريقة الثالثة

الوقت المستغرق لتنفيذ العمليات المحورية –بالنسبة إلى المثال– هو0.37 ثانية، مضافاً إليه الوقت الذي يستغرقه المعالج الرابع (CPU4) ويقدر بـ(0.23) ثانية فيكون الوقت المستغرق لكل المعالجات هو:

$(0.37 + 0.23)*3 = 1.8$ sec

وعليه فإن عامل التسريع سيكون كبيراً، ومن الممكن حسابه عن طريق:

$$\text{التسريع } Sp = \frac{\text{وقت التنفيذ لـM1 باستخدام معالج واحد (Ts)}}{\text{وقت التنفيذ لـM باستخدام P من المعالجات (Tp)}}$$

التسريع = 2.65 / 1.8 = 1.47

إذن تكون الطريقة الثالثة أسرع من الطريقة الأولى وأبطأ من الطريقة الثانية؛ وذلك لأن m<n فإن (n=4) بينما (m=2) –بالنسبة إلى المثال (1)– ولكن نأخذ بنظر الاعتبار عدد المعالجات، إذ استخدمنا في الطريقة الثانية (6) معالجات. أما في الطريقة الثالثة؛ فقد كانت (4) معالجات فقط، بينما كان عدد المعالجات في الطريقة الأولى (3) معالجات دائماً.

5. مقارنة بين الطرائق المتوازية الثلاث الجديدة:

من خلال النتائج التي ظهرت نلاحظ إن هناك فروقاً بين الطرائق يمكن تلخيصها بالجدول الآتي:

الجدول (5): الفروق بين الطرائق المتوازية الثلاث الجديدة للطريقة الفردية العادية

	الطريقة الأولى	الطريقة الثانية	الطريقة الثالثة
1.	نحتاج إلى ثلاث معالجات	نحتاج إلى n+2 من المعالجات	نحتاج إلى m+2 من المعالجات
2.	غير مكلفة لأن المسائل كافة تحتاج إلى ثلاث معالجات.	تعد مكلفة لمسألة صغيرة ولهذا يفضل استخدامها في المسائل الكبيرة.	تعد مكلفة لمسألة صغيرة ولهذا يفضل استخدامها في المسائل الكبيرة.
3.	عندما تكون المسألة كبيرة تأخذ وقتاً كبيراً بالنسبة للطريقتين الاخريين.	عندما تكون n>m يكون وقت التنفيذ أقل من الطريقة الثالثة ولكن بعدد معالجات أكثر.	عندما تكون n≤m يكون وقت التنفيذ أقل من الطريقة الثانية ولكن بعدد معالجات أكثر.
4.	الحمل يكون على المعالجات بشكل متساوٍ تقريباً.	الحمل على المعالج الأول والأخير يكون كثيراً.	الحمل يكون على المعالج الأخير كثيراً.

المصادر

(1) الالوسي، د.أحمد صالح وعادل زينل البياتي (1989): **مقدمة في التحليل العددي**، *مطبعة التعليم العالي في الموصل.*

(2) سيفي، د. علي محمد صادق و د. ابتسام كمال الدين (1986): **مبادئ التحليل العددي**، *مديرية دار الكتب للطباعة والنشر،جامعة الموصل.*

(3) عبد الحبيب عبد الله أحمد مرشد (2000): "**تقصي خوارزميات عددية لحل المعادلات التفاضلية الاعتيادية الصلبة الملائمة للحاسبات المتوازية**"، *أطروحة دكتوراه، كلية العلوم، جامعة الموصل .*

(4) عبد ذياب جزاع (1986): **بحوث العمليات**، *وزارة التعليم العالي والبحث العلمي، جامعة بغداد، الطبعة الثانية.*

(5) يحيى قاسم إبراهيم القاضلي (2002م): "**حل مسألة التخصيص باستخدام خوارزمية متوازية**"، *رسالة ماجستير ، كلية علوم الحاسبات والرياضيات، جامعة الموصل.*

[6] Bashir M. S., Khalaf and Khlil K.Abbo (2001): "Parallel Revised Simplex Method", ***Raf. Jour. Sci.**, Vol. 13, No. 1, pp51-60.*

[7] Bunday B. D. and Garside (1987): "**Linear Programming in Pascal**", *Edwared Arnold, London, U.K.*

[8] Flynn M. J. (1972): "Some Computer Organization and their Effectiveness". ***IEEE Transaction on /computers**, Vol. C-21, pp. 948-960.*

[9] Gavriel Yarmish & Richard Van Slyke (2001): "**An Implementation of the Standard Simplex Method**", *Technical Report, TE-CIS-2001-05*.

[10] http://ouray.cudenver.edu/~sfhopkin/, "**CSC 5809 Parallel and Distributed Systems", *Stephen Hopkins*.

[11] http://www.cs.bath.ac.uk/~amb/UQC007H3/ARCHITEC/ ARCH400. HTM

[12] http://www.krellinst.org/UCES/archive/classes/ASC/section2_6_1. html, MACS 440/563: "**Parallel Computing for Scientists and Engineers"**, *Manavendra Misra*.

[13] http://WWW.personal.rdg.ac.Uk/~sswills/history.html.

[14] http://www.ualberta.ca/CNS/hyperdispatch/HyperDispatch1/SP2Accounts.html

[15] http://www-unix.mcs.anl.gov/dbpp/text/node8.html, "**Designing and Building Parallel Programs**", *Ian Foster*.

[16] http://www. math. washington. edu / ~king / coursedir / m308a01, **Linear programming and Simplex Method**, Chapter 11.